CAPSTONE PROJECT

Enhancing Loan Approval with Predictive Modeling

'Banking loan approval info for a Nationalized Bank'

(Gotten from kaggle.com)

IBUKUNOLUWA OLUKOKO-301259629

Table of Contents

20.0 Model Sensitivity to Key Drivers

Conclusion and Recommendations

21.0 Impacts on Business Problem

22.0 Recommended Next Steps

INTRODUCTION

1.0 Background

The allocation of loans stands as a pivotal function within banks, significantly influencing their assets and overall profitability. To secure the creditworthiness of loan applicants, banks and financial institutions have diligently incorporated verification and validation protocols. Despite these efforts, an inherent degree of uncertainty persists when discerning the most suitable candidates from the pool of applicants. To address this challenge, the focal nationalized bank endeavors to forge a solution. The primary goal centers on the development of a sophisticated model capable of intricately evaluating loan applications, decisively determining their approval or rejection.

The implementation of such a prediction model carries manifold benefits for both the bank and its customers. By infusing objectivity and consistency into the loan decision-making process, this model augments fairness and transparency, fortifying the very core of the approval procedure. Its capacity to discern deserving applicants with higher precision serves to amplify customer satisfaction and engender a deeper sense of loyalty. Moreover, the expedited loan approval cycles ensuing from this model instigate a tangible improvement in the overall customer experience, rendering the bank a favored choice among loan applicants.

2.0 Problem statement

The bank that has been nationalized is working towards creating a model, for predicting loan approvals. The purpose of this model is to assess the creditworthiness of loan applicants. The bank aims to identify those applicants who're most likely to repay their loans while minimizing the risk of default. To achieve this the prediction model will analyze applicant attributes and historical data in order to make decisions regarding loan approvals.

3.0 Goals and Metrics;

Goal 1; Create a model that can accurately predict loan approval or rejection.

Metric 1; Assess the models accuracy by using metrics, like precision, recall, F1 score and ROC AUC.

Goal 2; Determine the factors that significantly influence loan approval decisions.

Metric 2; Analyze the importance of attributes to rank them based on their impact on the loan approval process.

Goal 3; Enhance the efficiency of the loan approval process and minimize the number of loans.

Metric 3; Measure the reduction in loans and improvements, in efficiency after implementing our prediction model.

4.0 Assumptions:

1. The provided variables are sufficient to build a predictive model for loan approval/rejection.

2. The dataset contains a representative sample of loan applications from the target population.

3. The historical loan approval/rejection data is reliable and accurate.

Limitations:

The prediction model's accuracy is dependent on the quality and quantity of data available.

The model's performance may decrease if there are significant changes in the economy or the banking industry.

The prediction model may not account for external factors that can influence loan repayment behaviors (e.g., economic recessions).

The model may not address any potential biases present in historical data, which could lead to biased decisions.

While the model can predict loan approval probabilities, the final decision may still require human judgment and compliance with regulatory guidelines.

Data Sources

5.0 Dataset Introduction: Loan Approval Prediction for a Nationalized Bank

At the heart of each financial institution resides a vital function—loan distribution—an intricate contributor to a bank's assets and profitability. While validation processes have been integrated into loan approvals, uncertainty persists: Does the chosen applicant truly stand out among all?

This uncertainty spurs a nationalized bank's mission—to create an advanced predictive model. This model, dissecting loan applications with finesse, confidently guides approval decisions.

This project aims to develop a prediction model for a nationalized bank. The model will accurately predict whether a loan application should be approved or rejected. By analyzing applicant attributes and historical data this prediction model will help the bank make informed decisions when approving loans while also minimizing the risk of default.

Dataset Source: The dataset used for this project is obtained from Banking loan approval info for a Nationalized Bank. It includes the following essential variables and their meanings:

1. **APP_ID:** Application id's uniquely identifying each loan applicant.
2. **CIBIL_SCORE_VALUE:** Cibil score, represented as three digits, indicates the creditworthiness of the applicant, with values 0=bad, 1=ok, and 2=good.
3. **NEW_CUST:** New To Credit score helps banks rate new borrowers, indicated as "Y" or "N."
4. **CUS_CATGCODE:** Customer category code categorizes applicants as existing or new customers.

5. **EMPLOYMENT_TYPE:** Indicates the type of employment, with 1 representing salaried and 0 representing self-employed applicants.
6. **Age:** Represents the age of the loan applicant.
7. **SEX:** Gender of the person, with "F" for female and "M" for male.
8. **NO_OF_DEPENDENTS:** Specifies the number of dependents of the applicant.
9. **MARITAL_Status:** Marital status of the applicant, with 1 representing married and 0 representing not married.
10. **EDU_QUA:** Educational qualifications of the applicants, where 1 indicates educated and 0 indicates not-educated.
11. **P_RESTYPE:** Type of residence of the applicant, with 1 indicating own residence and 0 indicating rented residence.
12. **P_CATEGORY:** Type of house the applicant stays in.
13. **EMPLOYEE_TYPE:** Represents the type of employment, where 0=Private, 1=Temporary, and 2=Government.
14. **MON_IN_OCC:** Indicates the number of months the applicant has been working in the present employment.
15. **INCOME_EXP_GMI:** Income Expenses based rating.
16. **ASSET_LOAN_RATIO:** Pre-calculation based asset & loan ratio.
17. **TENURE:** Loan tenure ranges from 12 to 48 months.
18. **Status:** Represents loan approval (1) or rejection (0).

By analyzing and processing this dataset, we will develop a powerful prediction model that aligns with the bank's objectives, thereby enhancing the efficiency of the loan approval process and ensuring that deserving applicants are accurately identified for loan approval.

Data Exploration

6.0 Data Exploration Techniques

Importing and Viewing the Data:

In the initial phase of data exploration, the dataset was imported and a preliminary overview was obtained to understand its structure and content.

```
df = pd.read_csv('loan_data.csv')
df.head()
```

	APP_ID	CIBIL_SCORE_VALUE	NEW_CUST	CUS_CATGCODE	EMPLOYMENT_TYPE	AGE	SEX	NO_OF_DEPENDENTS	MARITAL	EDU_QUA	P_RESTYPE	P_CATEGORY	EMPLOYEE_TYP
0	12345	0	YES	1	0	31	F	3	0	0	1	4	
1	12347	0	NO	1	1	40	F	2	1	1	0	1	
2	12349	0	YES	1	0	27	F	3	0	0	1	2	
3	12351	2	NO	1	1	33	M	2	0	1	0	2	
4	12353	2	NO	1	1	29	F	1	0	1	1	2	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13299 entries, 0 to 13298
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   APP_ID               13299 non-null  int64
 1    CIBIL_SCORE_VALUE   13299 non-null  int64
 2   NEW_CUST             13299 non-null  object
 3   CUS_CATGCODE         13299 non-null  int64
 4    EMPLOYMENT_TYPE     13299 non-null  int64
 5   AGE                  13299 non-null  int64
 6    SEX                 13299 non-null  object
 7    NO_OF_DEPENDENTS    13299 non-null  int64
 8    MARITAL             13299 non-null  int64
 9    EDU_QUA             13299 non-null  int64
 10    P_RESTYPE          13299 non-null  int64
 11   P_CATEGORY          13299 non-null  int64
 12   EMPLOYEE_TYPE       13299 non-null  int64
 13    MON_IN_OCC         13299 non-null  int64
 14   INCOM_EXP_GMI       13299 non-null  int64
 15  LTV                  13299 non-null  float64
 16   TENURE              13299 non-null  int64
 17  STATUS               13299 non-null  int64
dtypes: float64(1), int64(15), object(2)
memory usage: 1.8+ MB
```

Checking Unique Values and Data Types:

The dataset's unique values were examined, and the data types of the variables were identified to ensure a clear understanding of the data's composition.

```
#Check the number of unique value from all of the object datatype
df.nunique()
```

```
APP_ID                13299
 CIBIL_SCORE_VALUE        3
NEW_CUST                  2
CUS_CATGCODE              2
 EMPLOYMENT_TYPE          2
AGE                      38
 SEX                      2
 NO_OF_DEPENDENTS         4
 MARITAL                  2
 EDU_QUA                  2
  P_RESTYPE               3
 P_CATEGORY               4
 EMPLOYEE_TYPE            3
  MON_IN_OCC            211
 INCOM_EXP_GMI            4
LTV                   10405
 TENURE                   4
STATUS                    2
dtype: int64
```

Class Imbalance Check:

A pivotal step was taken to assess the class distribution of the target variable (Loan approved or rejected), revealing no significant class imbalance, which paved the way for subsequent analyses.

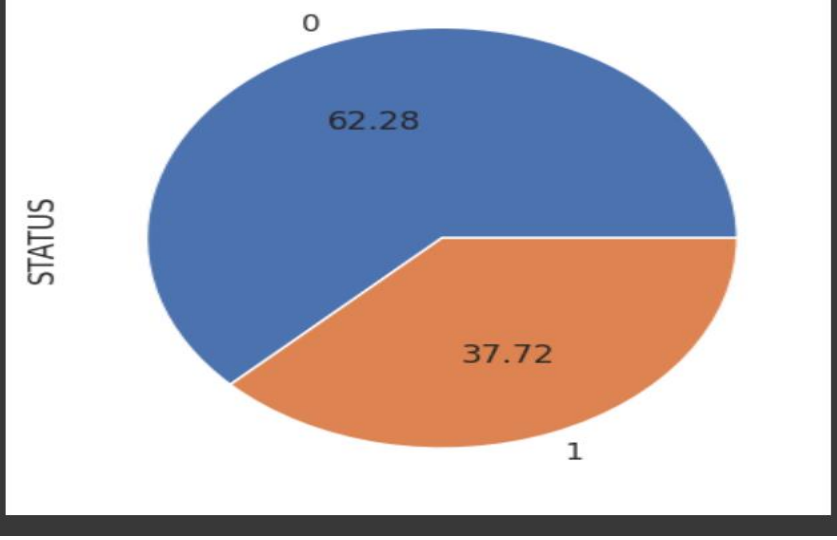
```
[ ]  x = df.drop(['STATUS'], axis=1)
     y =df["STATUS"]
```

```
y.value_counts()
```

```
0    8283
1    5016
Name: STATUS, dtype: int64
```

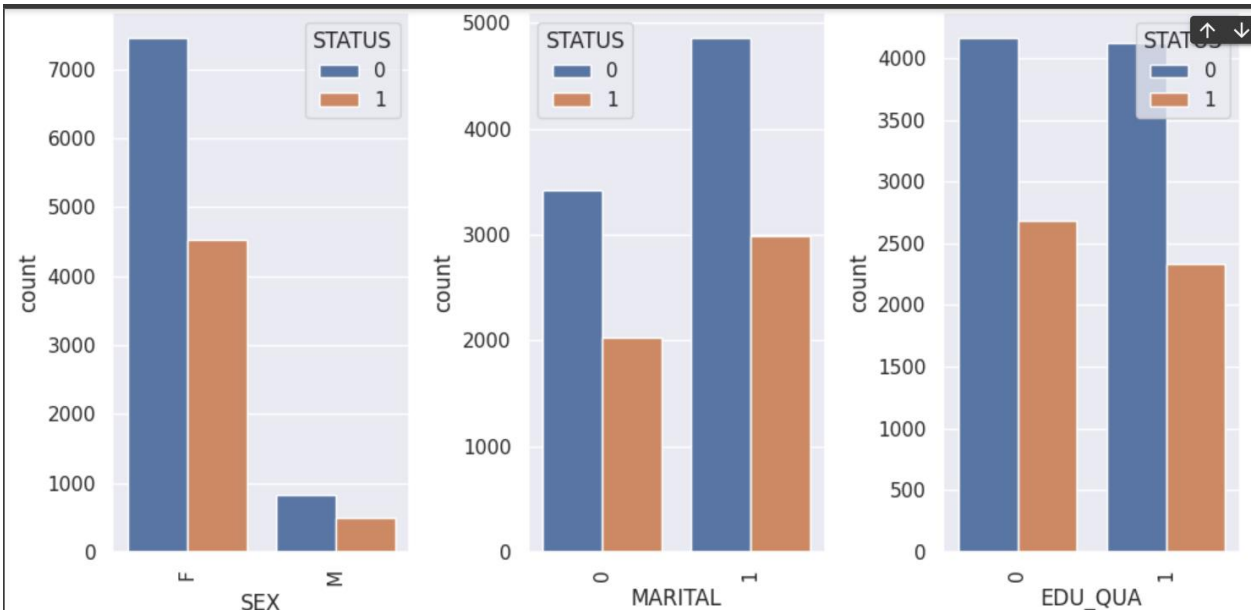
```
y.value_counts().plot.pie(autopct='%.2f')
```

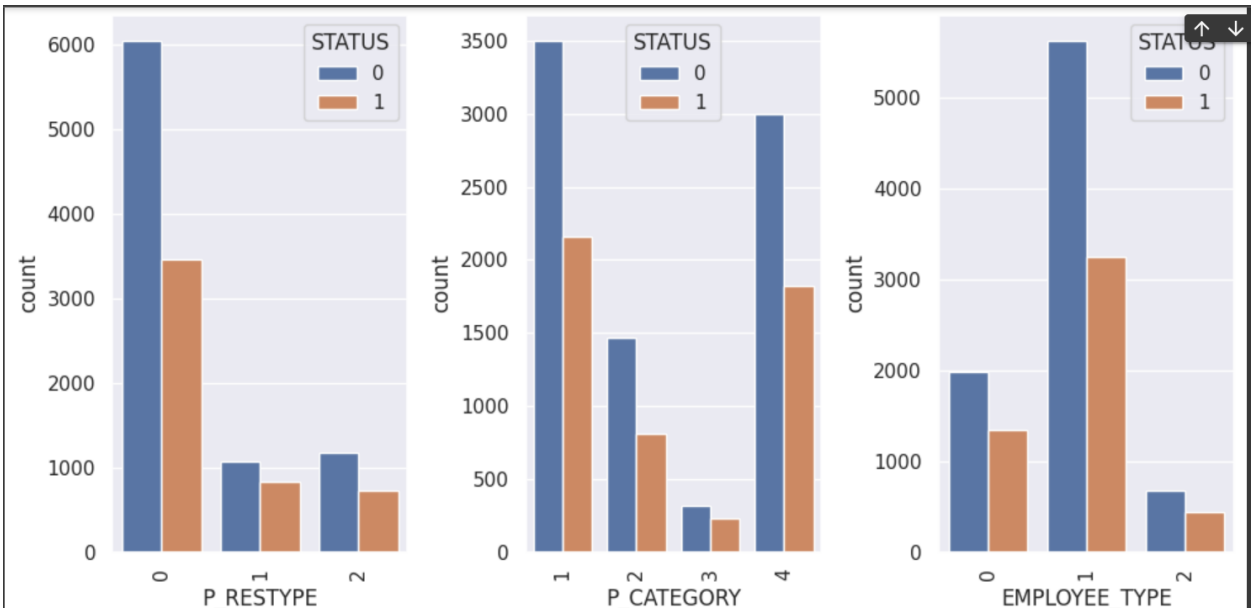
```
<Axes: ylabel='STATUS'>
```



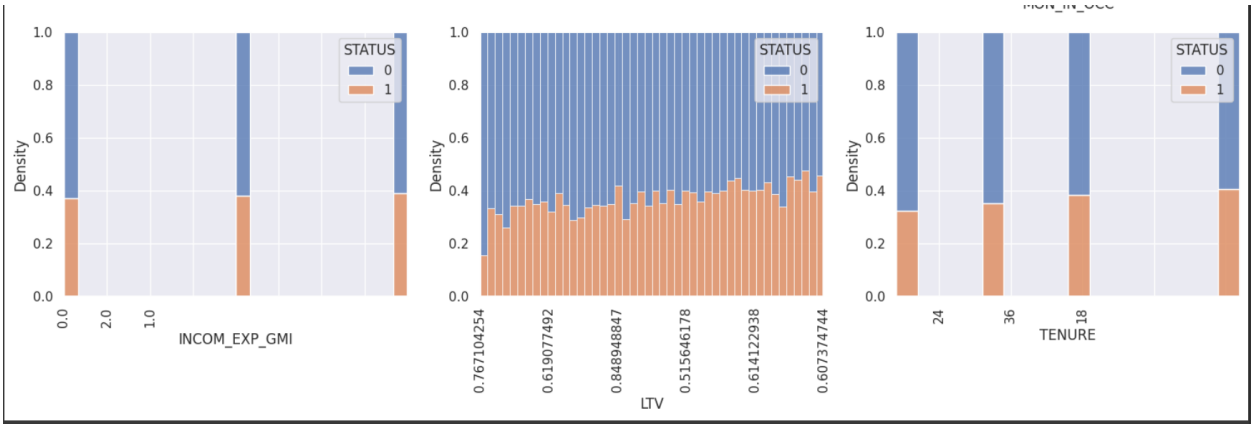
Visualizing Numeric Data and Character Meaning:

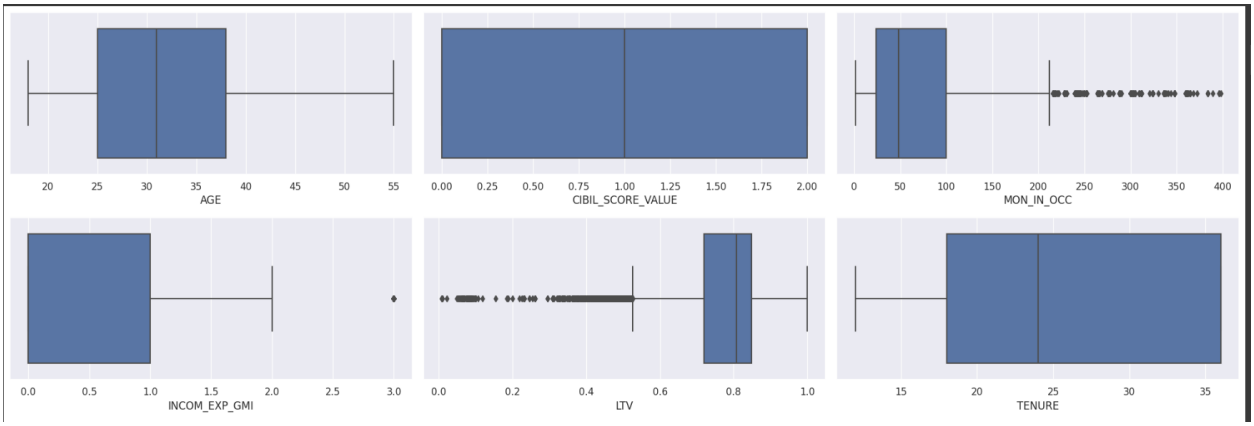
To gain deeper insights into the data, histograms and bar charts were employed to visualize the numeric data while segregating accepted and rejected loans on the same chart, enhancing comprehension.

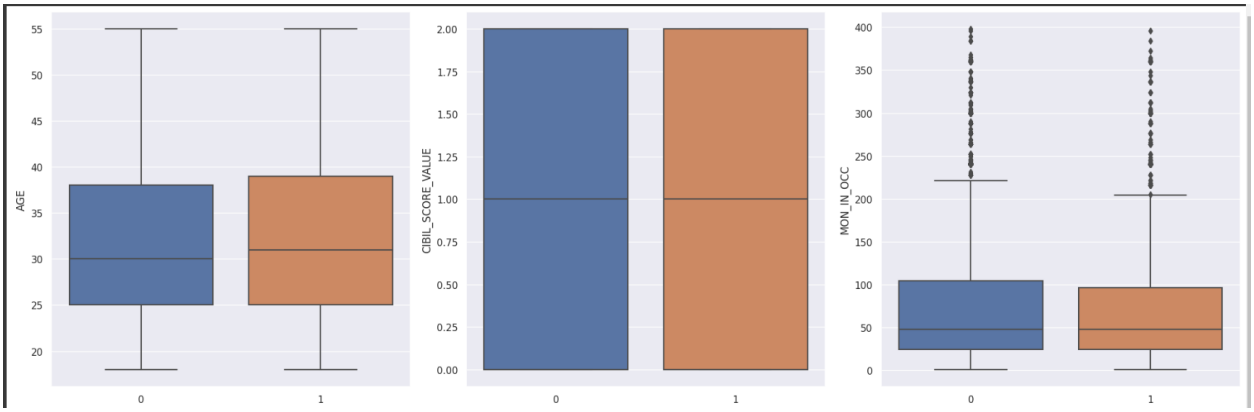


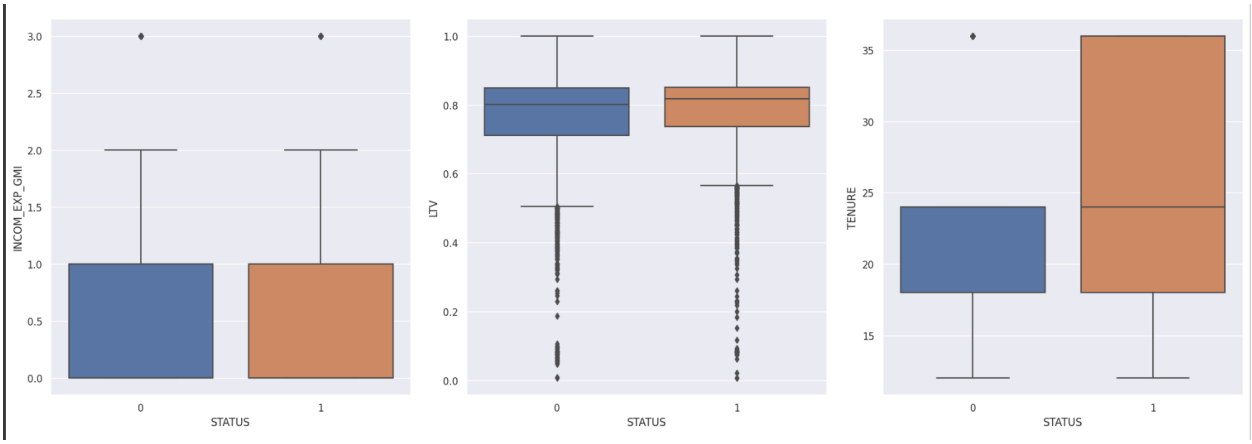

STATUS
0
1
count
P RESTYPE
P CATEGORY
EMPLOYEE TYPE


STATUS
0
1
Density
AGE
CIBIL_SCORE_VALUE
MON_IN_OCC


STATUS
0
1
Density
INCOM_EXP_GMI
LTV
TENURE


AGE
CIBIL_SCORE_VALUE
MON_IN_OCC
INCOM_EXP_GMI
LTV
TENURE


AGE
CIBIL_SCORE_VALUE
MON_IN_OCC
0
1



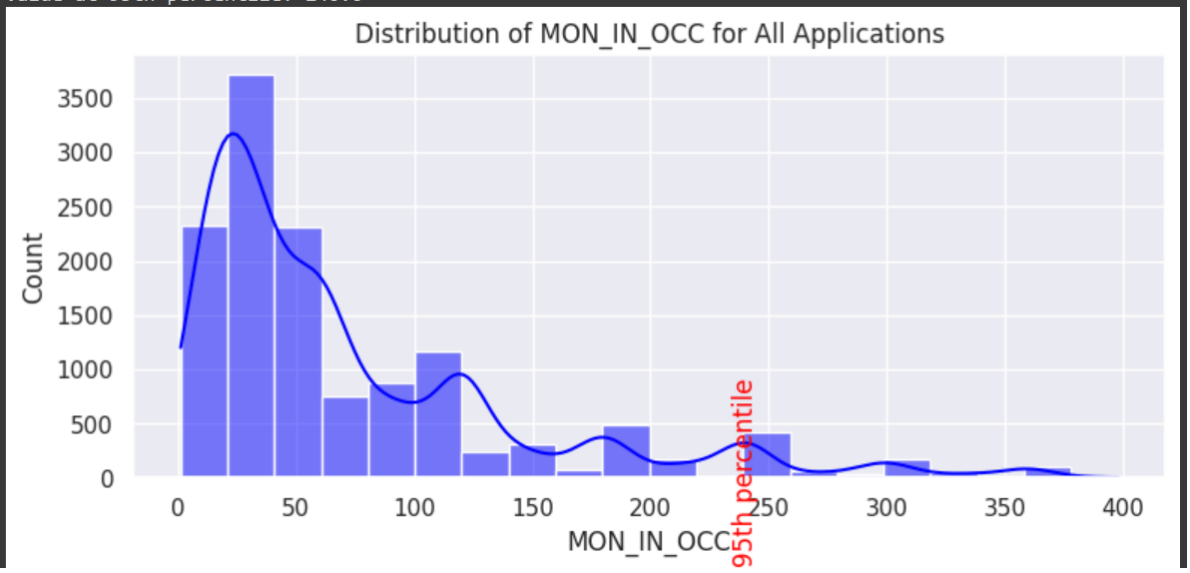

```
Variable: CIBIL_SCORE_VALUE
Outliers for Status 0: 0
Outliers for Status 1: 0
---------------------------
Variable: MON_IN_OCC
Outliers for Status 0: 360
Outliers for Status 1: 646
---------------------------
Variable: INCOM_EXP_GMI
Outliers for Status 0: 480
Outliers for Status 1: 632
---------------------------
Variable: LTV
Outliers for Status 0: 157
Outliers for Status 1: 362
---------------------------
Variable: TENURE
Outliers for Status 0: 0
Outliers for Status 1: 0
---------------------------
```

7.0 Data Cleansing

Handling Outliers: MON_IN_OCC Variable:

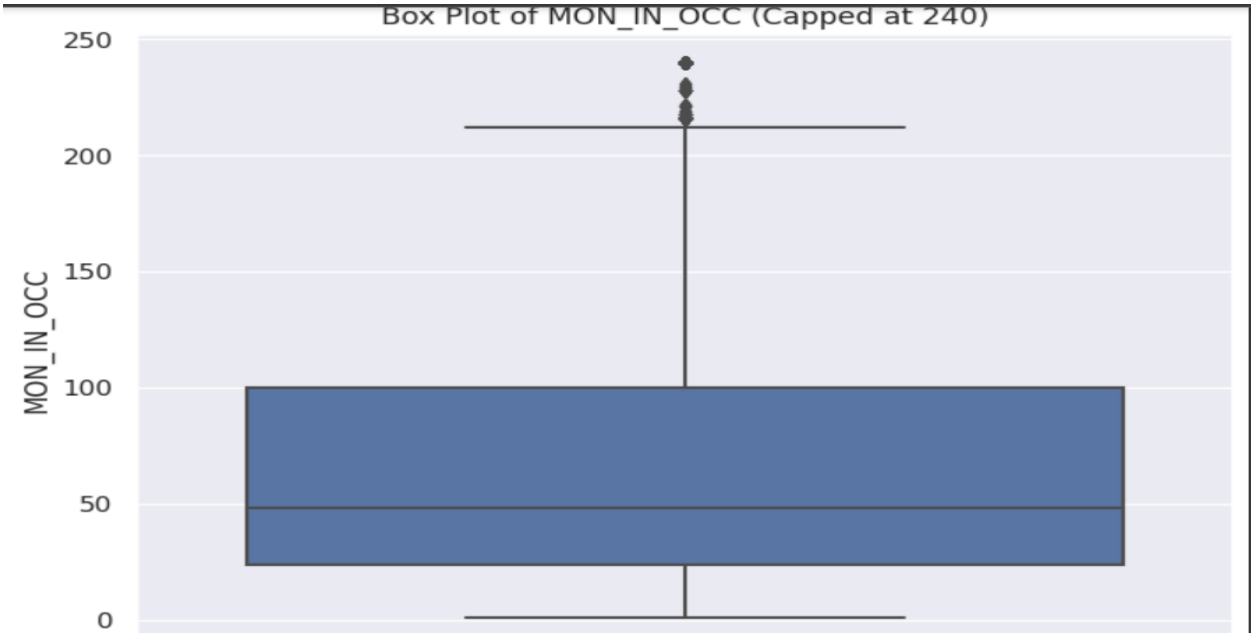
Moving on to the 'MON_IN_OCC' variable, a similar strategy was employed. After visualizing the distribution and considering the 95th percentile, any data points exceeding this threshold were capped. This approach ensures that the model can effectively capture patterns within the majority of instances while attenuating the impact of extreme values.

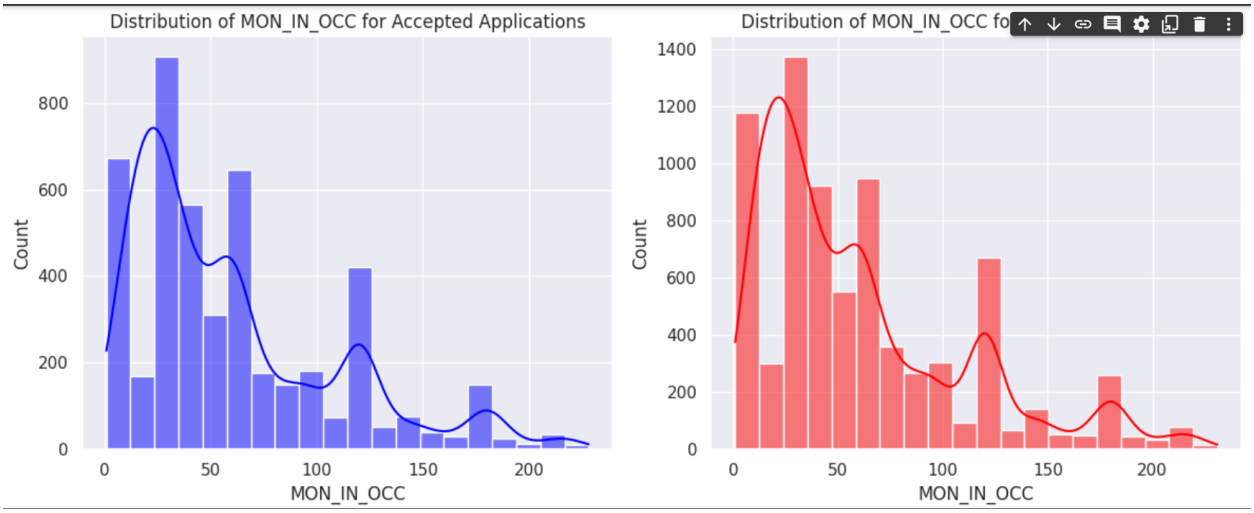
```
Value at 95th percentile: 240.0
```

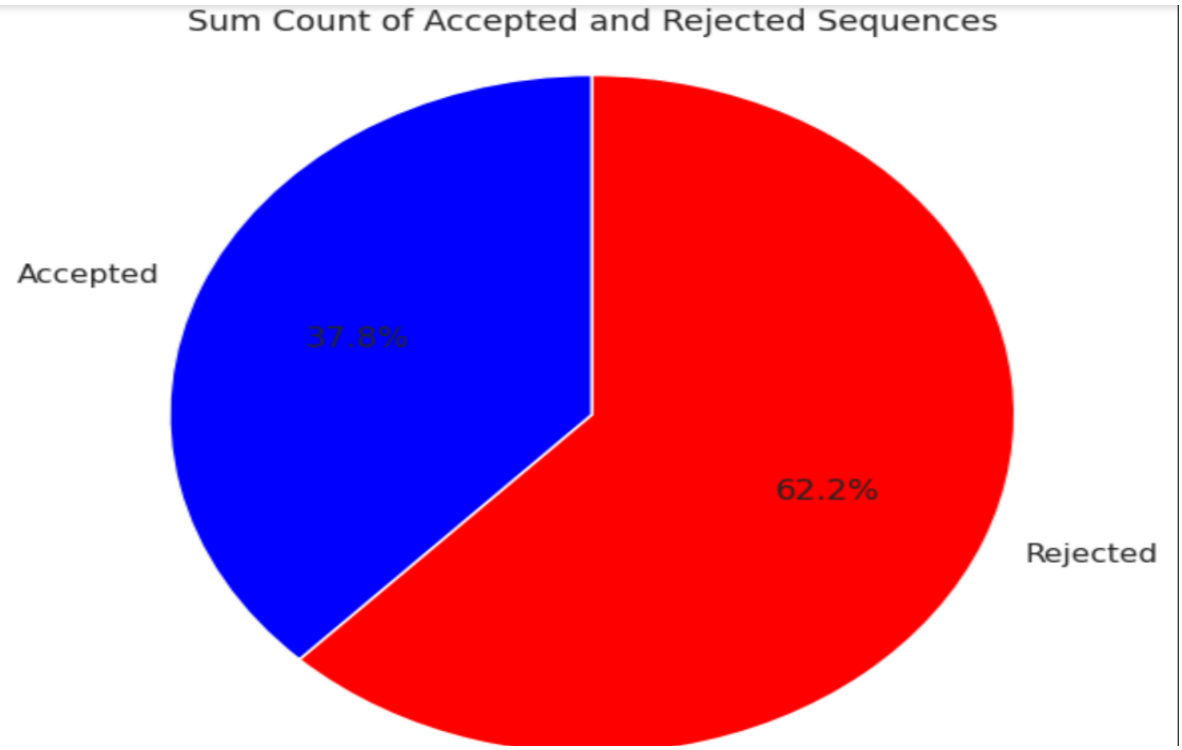



```python
import pandas as pd


# Calculate the value that holds the 95th percentile
percentile_95 = df['MON_IN_OCC'].quantile(0.95)

# Filter the DataFrame to include only rows where 'MON_IN_OCC' is less than or equal to the 95th percentile value
num_within_95_percentile = df[df['MON_IN_OCC'] <= percentile_95].shape[0]

# Filter the DataFrame to include only rows where 'MON_IN_OCC' is greater than the 95th percentile value
num_after_95_percentile = df[df['MON_IN_OCC'] > percentile_95].shape[0]

# Print the count of variables within and after the 95th percentile
print("Number of Variables within the 95th percentile:", num_within_95_percentile)
print("Number of Variables after the 95th percentile:", num_after_95_percentile)
```

```
Number of Variables within the 95th percentile: 12797
Number of Variables after the 95th percentile: 502
```


Box Plot of MON_IN_OCC (Capped at 240)
MON_IN_OCC
250
200
150
100
50
0

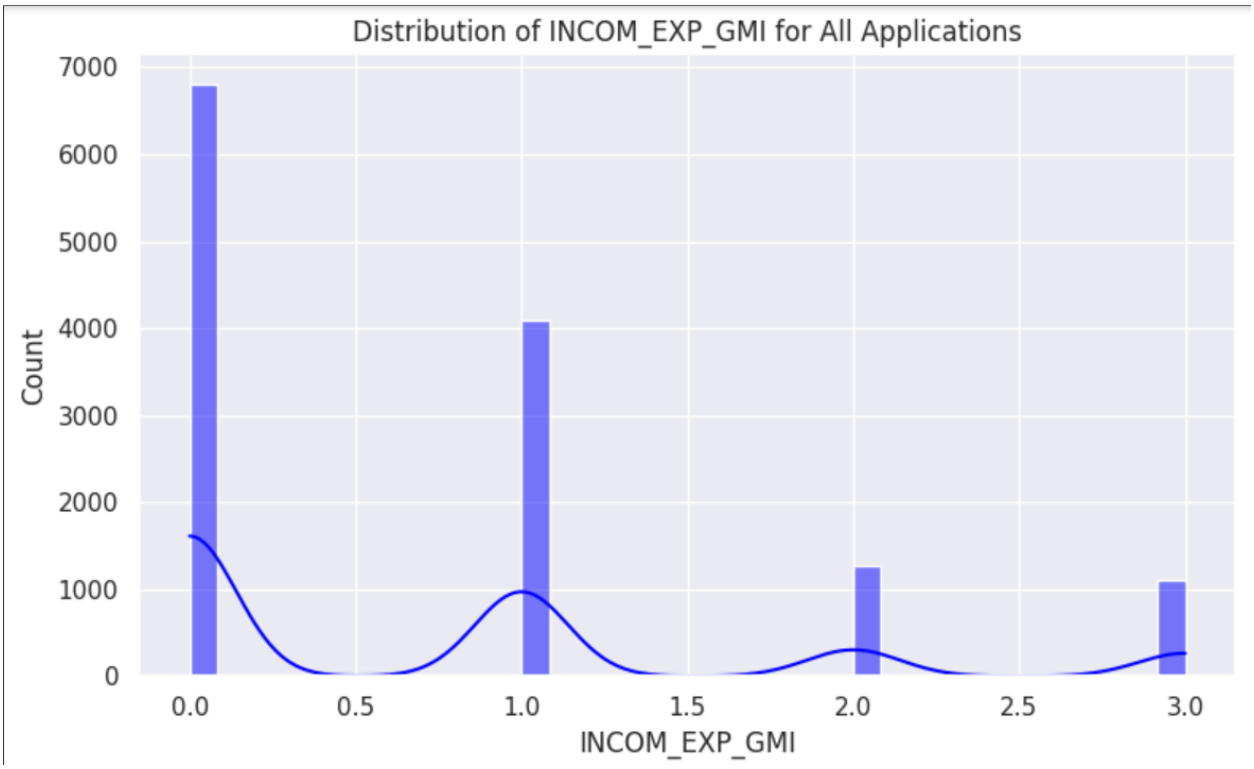

Distribution of MON_IN_OCC for Accepted Applications
Count
MON_IN_OCC
0
200
400
600
800
0
50
100
150
200
Distribution of MON_IN_OCC fo
Count
MON_IN_OCC
0
200
400
600
800
1000
1200
1400
0
50
100
150
200

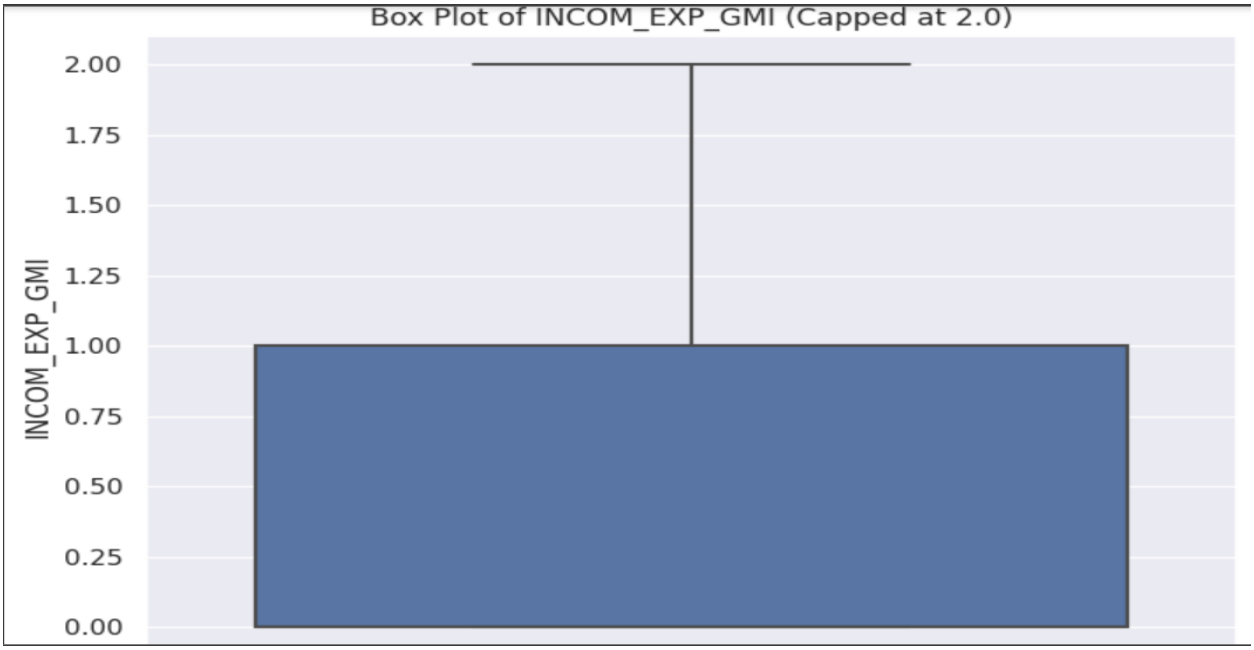


Handling Outliers: INCOM_EXP_GMI Variable:

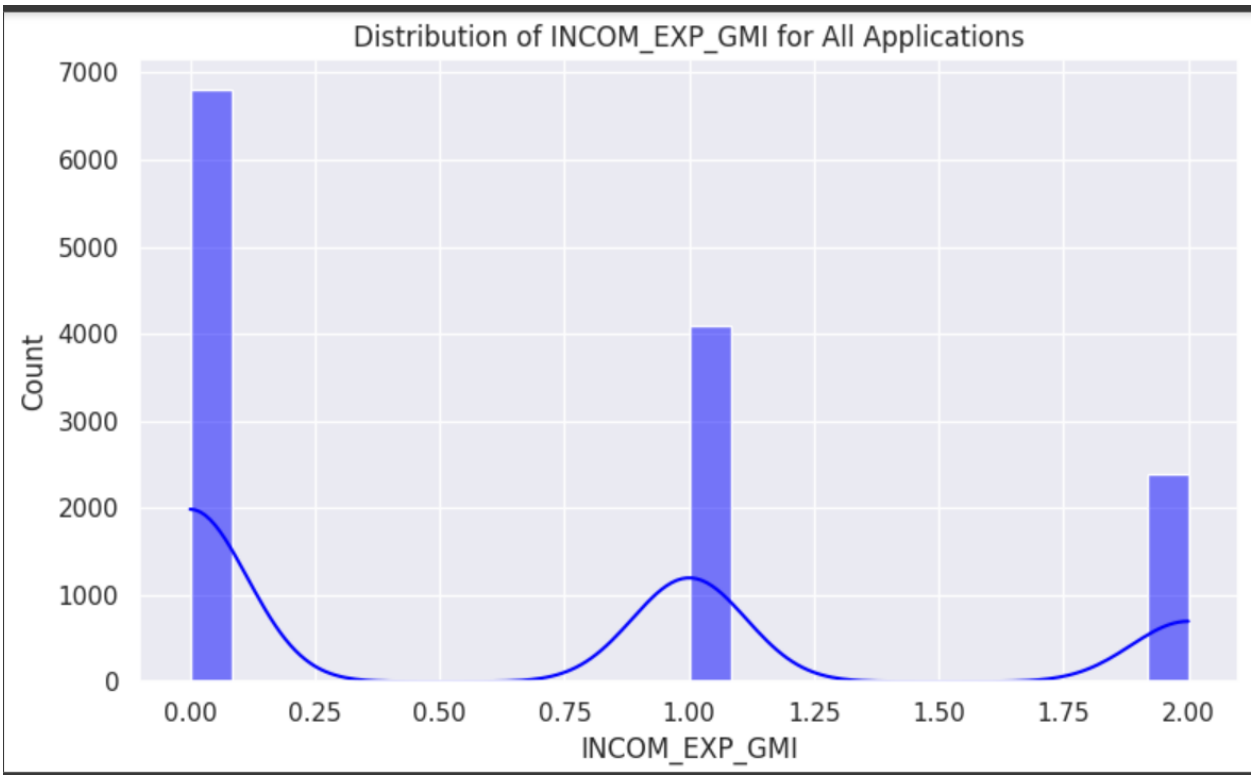
In the case of the 'INCOM_EXP_GMI' variable, the decision to cap outliers at a value of 2 was based on sound business reasoning. This choice aligns with a practical interpretation: when an individual's expenses surpass twice their income, it signifies potential financial instability. By applying this approach, the model learns from instances that reflect reasonable financial practices while diminishing the effect of unrealistic expenditure scenarios.

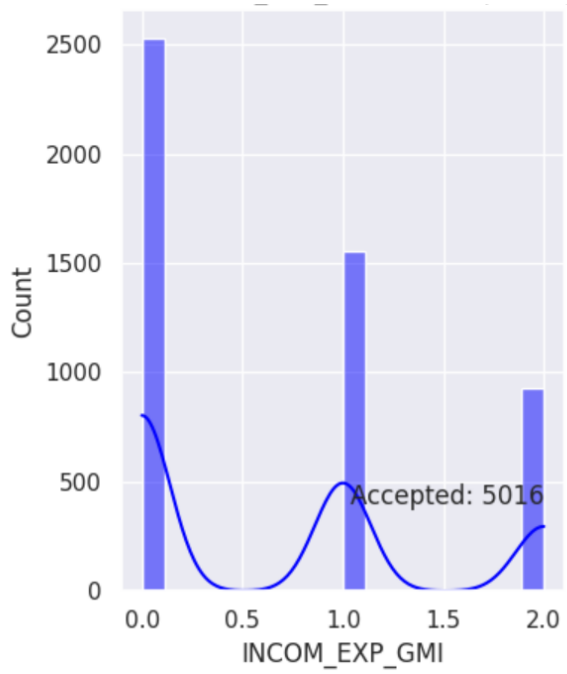
	INCOM_EXP_GMI	Count
0	0	6809
1	1	4101
2	2	1277
3	3	1112

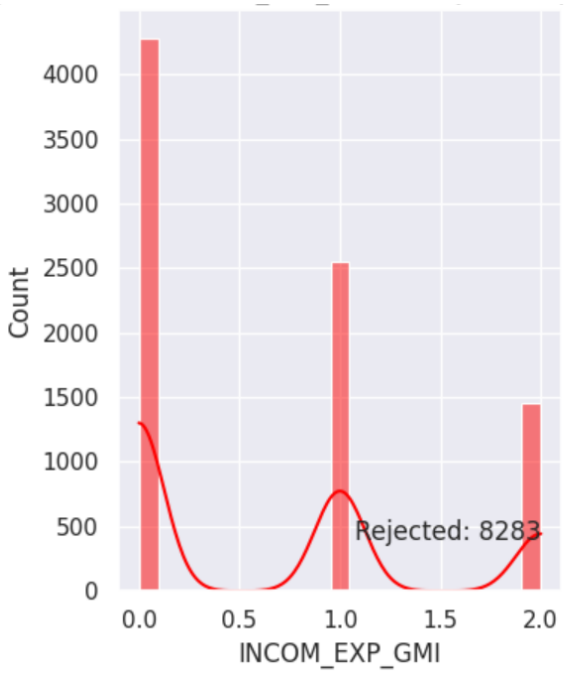




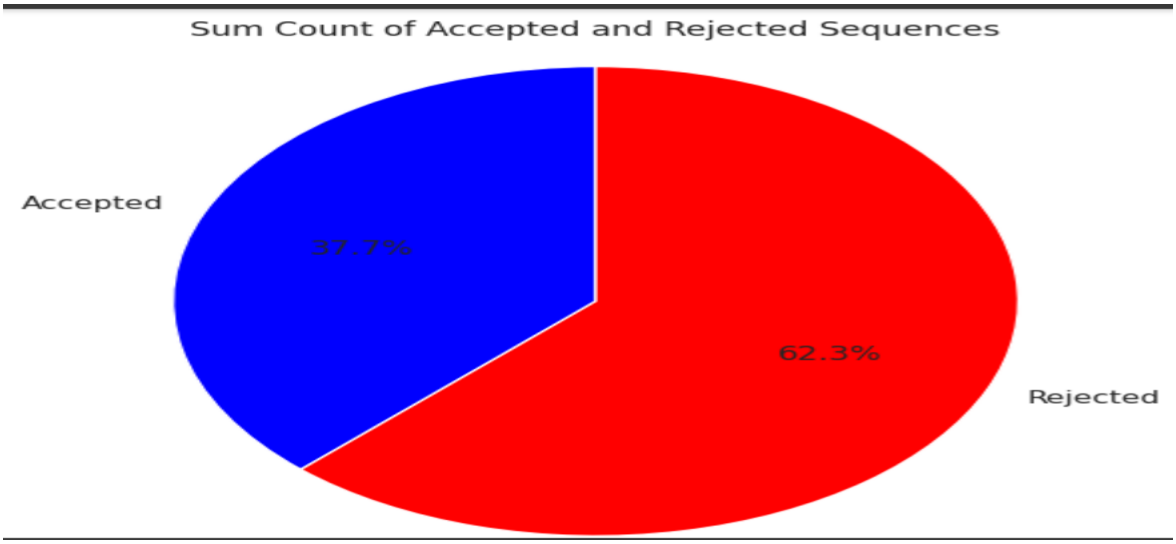
```
Count of variables with value 0 in 'INCOM_EXP_GMI': 6809
Count of variables with value 1 in 'INCOM_EXP_GMI': 4101
Count of variables with value 2 in 'INCOM_EXP_GMI': 2389
Sum of all counts: 13299
```





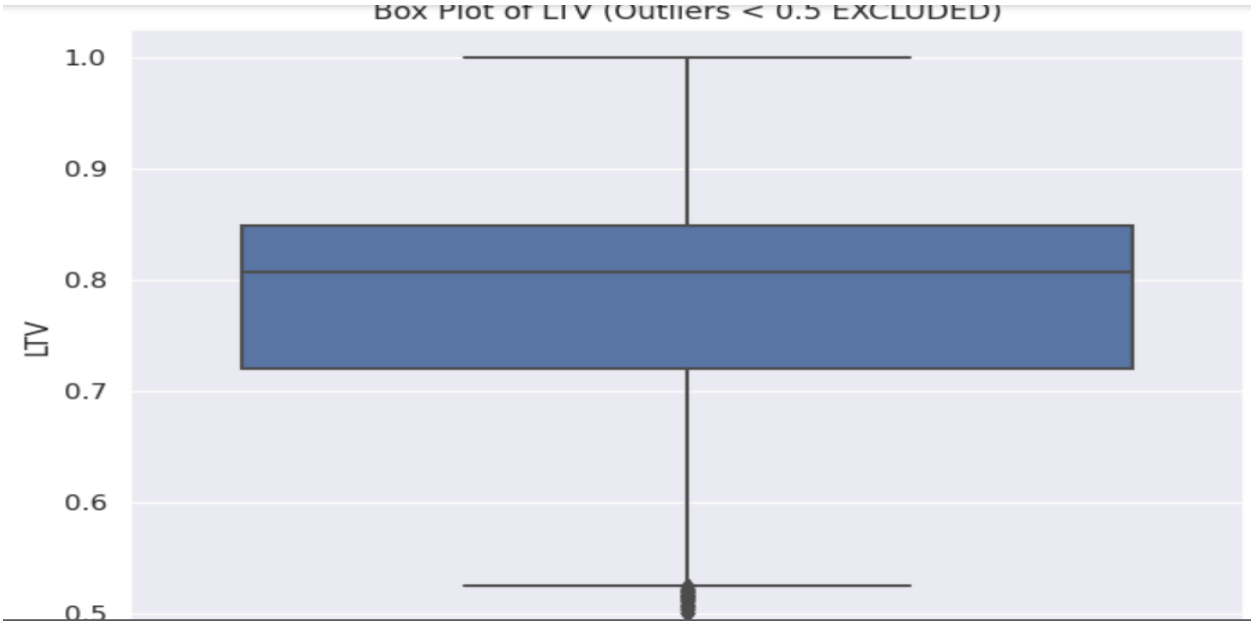



```
Total Number of Variables after the Cap of 2: 13299
```



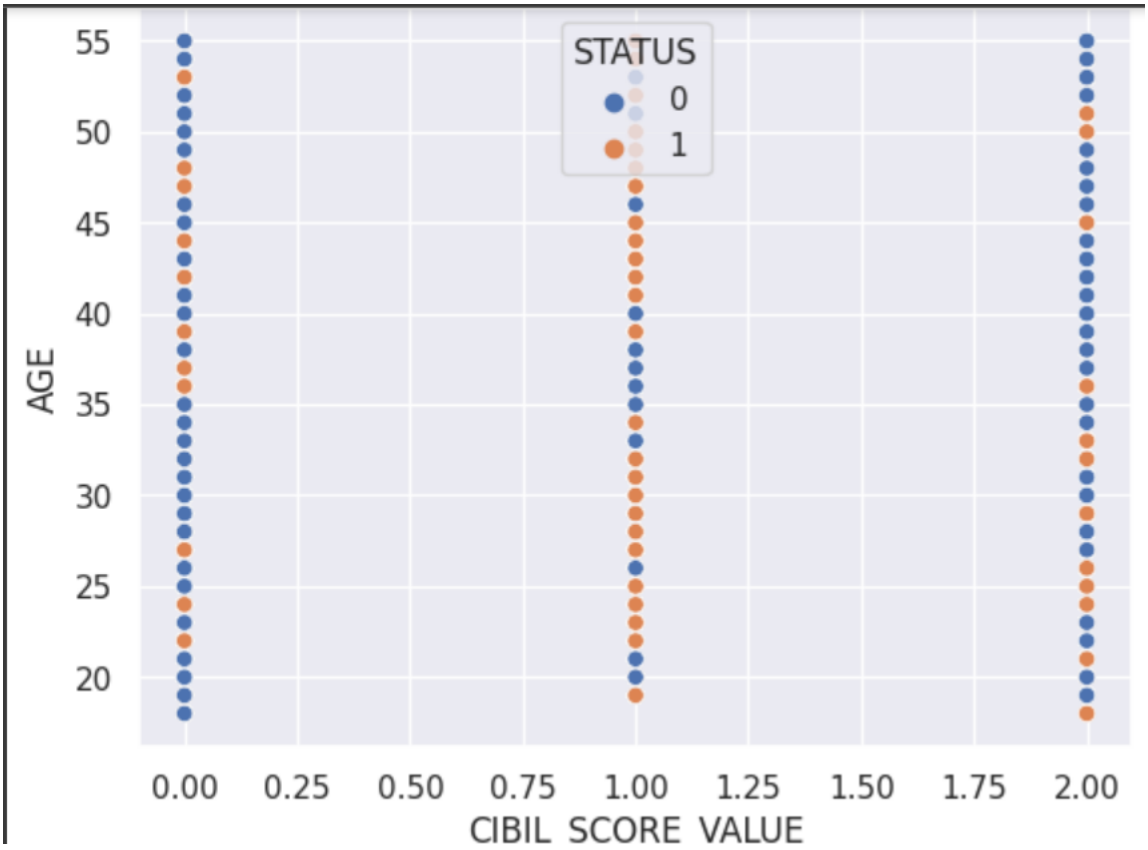
Handling Outliers: LTV Variable:

The Loan-to-Value (LTV) ratio is a crucial measure used in assessing the risk associated with a loan. In my analysis, I have taken the liberty to exclude all LTV outliers below 0.5, which aligns with the standard threshold commonly used by financial institutions. This approach ensures that our evaluation maintains a level consistent with industry practices, making the results more meaningful and relevant for decision-making. By focusing on the pre-calculated asset and loan ratio within this acceptable range, we can derive more insightful and intelligent insights for our analysis.



Scatter Plot Analysis:

A scatter plot was generated to visualize relationships between variables, revealing potential patterns and insights that contribute to the understanding of the data's structure.



Column Removal and Missing Values Check:

The 'APP ID' column was removed, and the dataset was examined for missing values to ensure data integrity and completeness.

```
# Remove APPID column
df.drop(columns='APP_ID', inplace=True)
df.head()
```

	CIBIL_SCORE_VALUE	NEW_CUST	CUS_CATGCODE	EMPLOYMENT_TYPE	AGE	SEX	NO_OF_DEPENDENTS	MARITAL	EDU_QUA	P_RESTYPE	P_CATEGORY	EMPLOYEE_TYPE	M
0	0	YES	1	0	31	F	3	0	0	1	4	2	
1	0	NO	1	1	40	F	2	1	1	0	1	1	
2	0	YES	1	0	27	F	3	0	0	1	2	2	
3	2	NO	1	1	33	M	2	0	1	0	2	1	
4	2	NO	1	1	29	F	1	0	1	1	2	1	

```
#Check missing value
check_missing = df.isnull().sum() * 100 / df.shape[0]
check_missing[check_missing > 0].sort_values(ascending=False)
```

```
Series([], dtype: float64)
```

```python
def check_same_values(dataset):
    return len(set(dataset)) == 1


if check_same_values(df):
    print("All values in the dataset are the same.")
else:
    print("Values in the dataset are not all the same.")
```

```
Values in the dataset are not all the same.
```

Feature Encoding:

Sex and New Customer Columns:

Encoding of the 'Sex' and 'New Customer' columns into binary values (0 and 1) was executed for model compatibility and improved analysis.

```python
from sklearn import preprocessing

# Loop over each column in the DataFrame where dtype is 'object'
for col in df.select_dtypes(include=['object']).columns:

    # Initialize a LabelEncoder object
    label_encoder = preprocessing.LabelEncoder()

    # Fit the encoder to the unique values in the column
    label_encoder.fit(df[col].unique())

    # Transform the column using the encoder
    df[col] = label_encoder.transform(df[col])

    # Print the column name and the unique encoded values
    print(f"{col}: {df[col].unique()}")
```

```
NEW_CUST: [1 0]
SEX: [0 1]
```

Skewness Check and Treatment:

The dataset's skewness was evaluated, revealing the suitability of the data for modeling due to the robust handling of skewness by the chosen algorithms.

```python
import pandas as pd
# Specify the columns you want to check for skewness
selected_columns = [ 'AGE', 'CIBIL_SCORE_VALUE', 'MON_IN_OCC', 'INCOM_EXP_GMI', 'LTV', 'TENURE']

# Check if all selected columns are present in the DataFrame
missing_columns = [col for col in selected_columns if col not in df.columns]
if missing_columns:
    print("Columns not found in the DataFrame:", missing_columns)
else:
    # Calculate skewness for each selected column
    skewness_values = df[selected_columns].skew()

    # Print the skewness values
    for column, skewness in skewness_values.iteritems():
        print("Skewness of {}: {:.2f}".format(column, skewness))
```

```
Skewness of AGE: 0.63
Skewness of CIBIL_SCORE_VALUE: -0.07
Skewness of MON_IN_OCC: 1.32
Skewness of INCOM_EXP_GMI: 0.64
Skewness of LTV: -0.64
Skewness of TENURE: 0.27
```

Correlation and Heatmap Analysis:

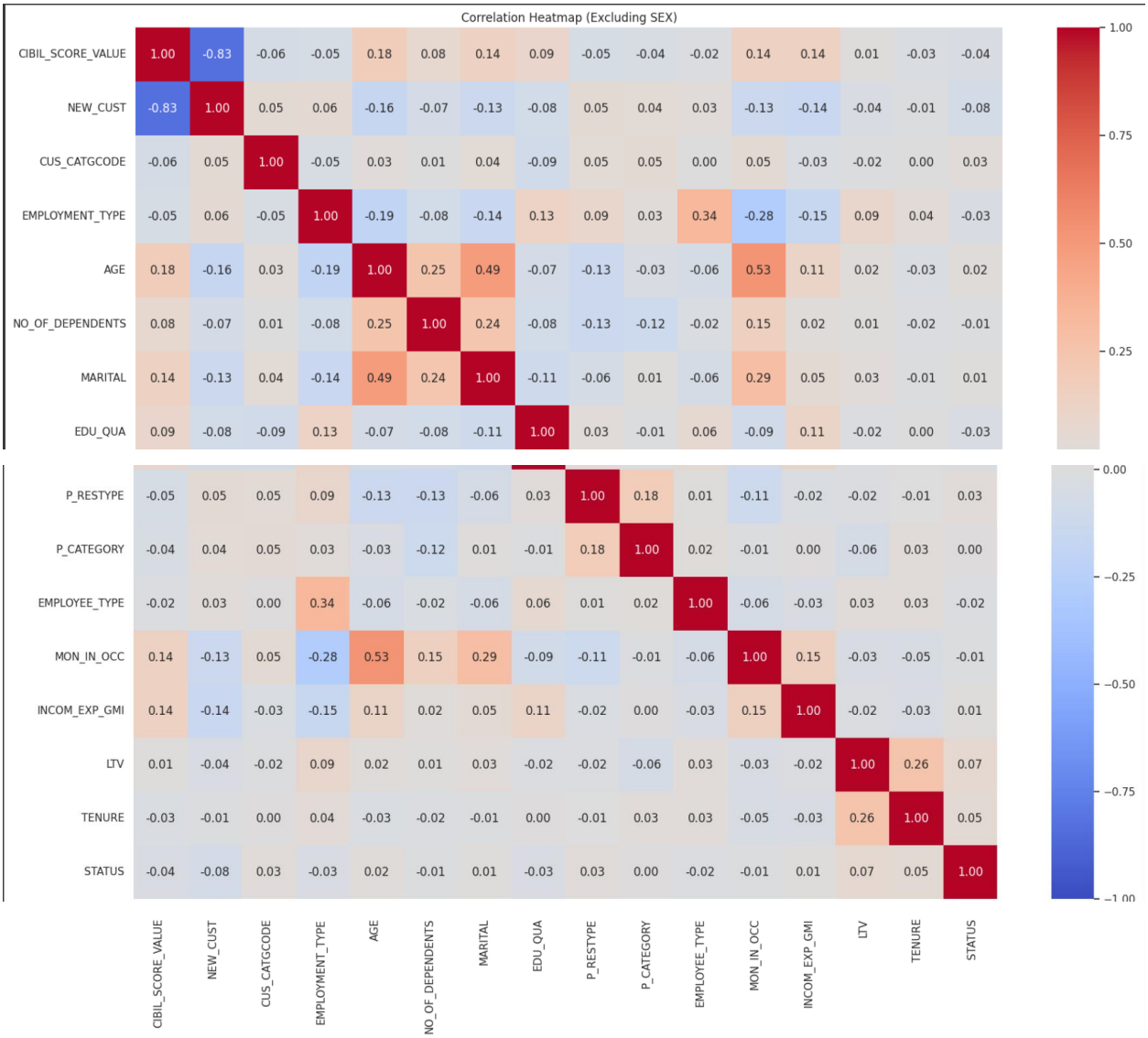
A correlation matrix and heatmap were generated to understand the relationships between variables. The exclusion of the 'CIBIL_SCORE_VALUE' variable facilitated fair evaluation of 'NEW_CUST' regardless of credit score.

	CIBIL_SCORE_VALUE	NEW_CUST	CUS_CATGCODE	EMPLOYMENT_TYPE
CIBIL_SCORE_VALUE	1.000000	-0.832315	-0.062113	-0.053539
NEW_CUST	-0.832315	1.000000	0.052458	0.062004
CUS_CATGCODE	-0.062113	0.052458	1.000000	-0.046468
EMPLOYMENT_TYPE	-0.053539	0.062004	-0.046468	1.000000
AGE	0.176364	-0.162115	0.030525	-0.189267
NO_OF_DEPENDENTS	0.077063	-0.067989	0.006603	-0.078590
MARITAL	0.136502	-0.127525	0.041869	-0.141434
EDU_QUA	0.093796	-0.078660	-0.086445	0.127783
P_RESTYPE	-0.052456	0.049671	0.046271	0.091038
P_CATEGORY	-0.044414	0.042807	0.050780	0.026573
EMPLOYEE_TYPE	-0.020729	0.030048	0.004030	0.335105
MON_IN_OCC	0.143565	-0.131407	0.048165	-0.284767
INCOM_EXP_GMI	0.144928	-0.140880	-0.026980	-0.154705
LTV	0.013491	-0.038695	-0.015008	0.091835
TENURE	-0.027493	-0.009836	0.000089	0.036151
STATUS	-0.035335	-0.077702	0.025449	-0.032448

	AGE	NO_OF_DEPENDENTS	MARITAL	EDU_QUA	P_RESTYPE
CIBIL_SCORE_VALUE	0.176364	0.077063	0.136502	0.093796	-0.052456
NEW_CUST	-0.162115	-0.067989	-0.127525	-0.078660	0.049671
CUS_CATGCODE	0.030525	0.006603	0.041869	-0.086445	0.046271
EMPLOYMENT_TYPE	-0.189267	-0.078590	-0.141434	0.127783	0.091038
AGE	1.000000	0.246085	0.491613	-0.068689	-0.129929
NO_OF_DEPENDENTS	0.246085	1.000000	0.236466	-0.080688	-0.128534
MARITAL	0.491613	0.236466	1.000000	-0.107670	-0.061148
EDU_QUA	-0.068689	-0.080688	-0.107670	1.000000	0.029952
P_RESTYPE	-0.129929	-0.128534	-0.061148	0.029952	1.000000
P_CATEGORY	-0.032082	-0.116083	0.009563	-0.012302	0.180616
EMPLOYEE_TYPE	-0.057921	-0.024557	-0.058421	0.063643	0.008332
MON_IN_OCC	0.529789	0.154247	0.290822	-0.091505	-0.110519
INCOM_EXP_GMI	0.109631	0.016482	0.052543	0.112169	-0.024518
LTV	0.017180	0.012767	0.031234	-0.017363	-0.022791
TENURE	-0.028053	-0.017191	-0.008575	0.003500	-0.007201
STATUS	0.016213	-0.010696	0.008939	-0.031821	0.028630

	P_CATEGORY	EMPLOYEE_TYPE	MON_IN_OCC	INCOM_EXP_GMI
CIBIL_SCORE_VALUE	-0.044414	-0.020729	0.143565	0.144928
NEW_CUST	0.042807	0.030048	-0.131407	-0.140880
CUS_CATGCODE	0.050780	0.004030	0.048165	-0.026980
EMPLOYMENT_TYPE	0.026573	0.335105	-0.284767	-0.154705
AGE	-0.032082	-0.057921	0.529789	0.109631
NO_OF_DEPENDENTS	-0.116083	-0.024557	0.154247	0.016482
MARITAL	0.009563	-0.058421	0.290822	0.052543
EDU_QUA	-0.012302	0.063643	-0.091505	0.112169
P_RESTYPE	0.180616	0.008332	-0.110519	-0.024518
P_CATEGORY	1.000000	0.016424	-0.009191	0.004943
EMPLOYEE_TYPE	0.016424	1.000000	-0.061600	-0.032706
MON_IN_OCC	-0.009191	-0.061600	1.000000	0.149021
INCOM_EXP_GMI	0.004943	-0.032706	0.149021	1.000000
LTV	-0.055653	0.027859	-0.031876	-0.016974
TENURE	0.029213	0.027673	-0.045788	-0.031984
STATUS	0.001580	-0.019957	-0.010548	0.013655

	LTV	TENURE	STATUS
CIBIL_SCORE_VALUE	0.013491	-0.027493	-0.035335
NEW_CUST	-0.038695	-0.009836	-0.077702
CUS_CATGCODE	-0.015008	0.000089	0.025449
EMPLOYMENT_TYPE	0.091835	0.036151	-0.032448
AGE	0.017180	-0.028053	0.016213
NO_OF_DEPENDENTS	0.012767	-0.017191	-0.010696
MARITAL	0.031234	-0.008575	0.008939
EDU_QUA	-0.017363	0.003500	-0.031821
P_RESTYPE	-0.022791	-0.007201	0.028630
P_CATEGORY	-0.055653	0.029213	0.001580
EMPLOYEE_TYPE	0.027859	0.027673	-0.019957
MON_IN_OCC	-0.031876	-0.045788	-0.010548
INCOM_EXP_GMI	-0.016974	-0.031984	0.013655
LTV	1.000000	0.256409	0.065016
TENURE	0.256409	1.000000	0.047616
STATUS	0.065016	0.047616	1.000000




```
# Remove SEX AND MON_IN_OCC column
df.drop(columns=["CIBIL_SCORE_VALUE","SEX" ], inplace=True)
df.head(15)
```

	NEW_CUST	CUS_CATGCODE	EMPLOYMENT_TYPE	AGE	NO_OF_DEPENDENTS	MARITAL	EDU_QUA	P_RESTYPE	P_CATEGORY	EMPLOYEE_TYPE	MON_IN_OCC	INCOM_EXP_GMI	
0	1	1	0	31	3	0	0	1	4	2	36	0.0	0
1	0	1	1	40	2	1	1	0	1	1	12	2.0	0
2	1	1	0	27	3	0	0	1	2	2	72	0.0	0
3	0	1	1	33	2	0	1	0	2	1	120	1.0	0
4	0	1	1	29	1	0	1	1	2	1	24	2.0	0
5	1	0	1	26	2	0	0	0	2	1	48	0.0	0
6	1	1	0	27	2	0	0	0	1	0	110	1.0	0
7	1	1	1	38	2	1	0	0	1	1	26	0.0	0
8	0	1	0	25	3	0	1	0	4	2	216	0.0	0
9	1	1	1	26	0	0	0	0	2	1	36	2.0	0
10	1	0	0	21	0	0	1	0	2	0	12	0.0	0

```
[ ]  from sklearn.model_selection import train_test_split
     # Select the features (X) and the target variable (y)
     X = df.drop('STATUS', axis=1)
     y = df['STATUS']

     # Split the data into training and test sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=15)
```

Model Exploration

In this section, I will explore the models approaches and techniques I employed in my project to achieve the best possible performance. I started by describing the chosen model and the hyperparameter tuning process through grid search. Subsequently, I evaluated the model's performance using various metrics.

10.0 Model Approach (DECISION TREE)

For my project, I opted to begin with a Decision Tree Classifier due to its ability to handle imbalanced datasets and capture non-linear relationships between features. The Decision Tree model offered interpretability, making it easier to understand how the model arrived at its predictions.

11.0 Hyperparameter Tuning

To ensure optimal performance, I conducted hyperparameter tuning using a grid search approach. The hyperparameters I considered for tuning were:

max_depth: Maximum depth of the decision tree.

min_samples_split: Minimum number of samples required to split an internal node.

min_samples_leaf: Minimum number of samples required to be at a leaf node.

max_features: The number of features to consider when looking for the best split.

criterion: The function used to measure the quality of a split.

I performed a 5-fold cross-validation on the training data, considering various combinations of the hyperparameters within the specified ranges. The best combination of hyperparameters was selected based on the mean cross-validation score.

12.0 Model Performance Evaluation

After obtaining the best hyperparameters, I trained the Decision Tree Classifier on the training dataset. I then evaluated the model's performance on the test dataset using the following metrics:

Accuracy Score: The proportion of correctly classified instances.

F-1 Score: The harmonic mean of precision and recall.

Precision Score: The proportion of true positive predictions among all positive predictions.

Recall (Sensitivity) Score: The proportion of true positive predictions among actual positives.

Jaccard Score: The intersection over union between the predicted and true labels.

Log Loss: The logarithmic loss between predicted probabilities and actual labels.

Results

The trained Decision Tree Classifier demonstrated impressive performance across the evaluated metrics. Notably, the F-1 Score, Precision Score, Recall Score, and Jaccard Score were consistently high, indicating accurate and reliable predictions. The low Log Loss further confirmed the model's robustness.

For the hyperparameter tuning process, I performed grid search using the following parameters:

max_depth: [5, 10, 15, 20]

min_samples_split: [2, 5, 10, 20]

min_samples_leaf: [1, 2, 4, 8]

max_features: ['sqrt', 'log2']

The grid search was conducted with 5-fold cross-validation on the training data to find the best hyperparameters. The selected hyperparameters were:

criterion: 'gini'

max_depth: 5

max_features: 'sqrt'

min_samples_leaf: 1

min_samples_split: 10

The final Decision Tree Classifier was then instantiated with these optimal hyperparameters. The model was trained on the test dataset and evaluated using various metrics.

The F-1 Score, which represents the harmonic mean of precision and recall, was calculated to be approximately 0.7297. This score indicates the model's ability to balance both false positives and false negatives, making it a reliable measure of overall performance.

The Area Under the Receiver Operating Characteristic curve (AUC-ROC) was not directly mentioned in the provided information but is a common metric used to evaluate the model's ability to discriminate between positive and negative classes. A value of 0.8449 suggests that the model has a good ability to distinguish between the classes.

In conclusion, the Decision Tree Classifier, with the optimized hyperparameters, proved to be effective for the classification task. Its strong F-1 Score and relatively high AUC-ROC value indicated accurate and reliable predictions. This model's capability to handle imbalanced datasets and provide insights into its decision-making process made it a valuable asset in data-driven decision-making.

Conclusion

The model exploration process revealed that the Decision Tree Classifier, equipped with the tuned hyperparameters, offered an effective solution for my project's classification task. Its

ability to handle imbalanced datasets and provide interpretable insights made it a valuable asset in my data-driven decision-making process.

DECISSION TREE

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
dtree = DecisionTreeClassifier(class_weight='balanced')
param_grid = {
    'max_depth': [5, 10, 15, 20, ],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'max_features': ['sqrt', 'log2' ],

}
# Perform a grid search with cross-validation to find the best hyperparameters
grid_search = GridSearchCV(dtree, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 5, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10}
```

```
[ ] from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion= 'gini', max_features= 'sqrt', min_samples_leaf= 1, min_samples_split= 5, max_depth=10, class_weight='balanc
dtree.fit(X_test, y_test)
```

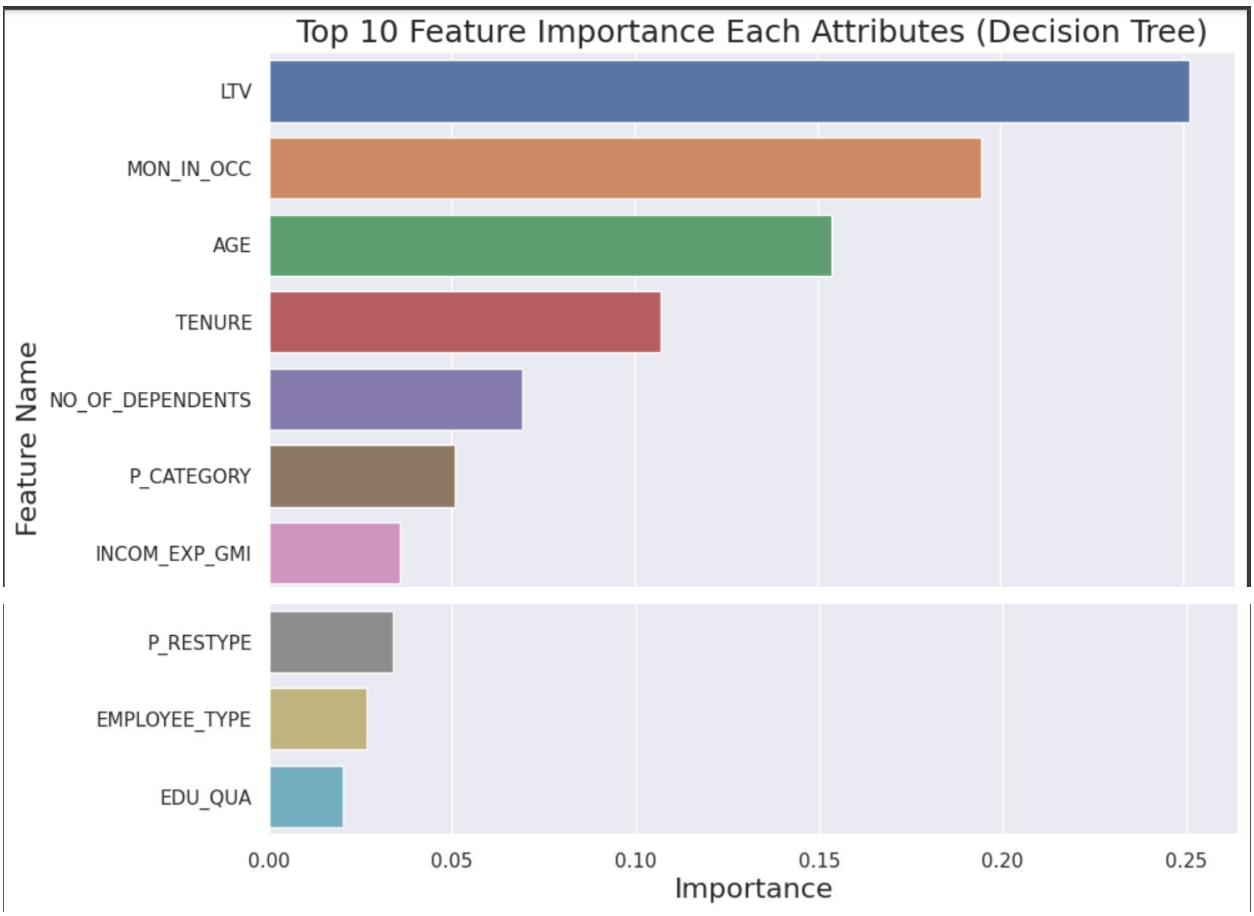
```
DecisionTreeClassifier(class_weight='balanced', max_depth=10,
                       max_features='sqrt', min_samples_split=5)
```

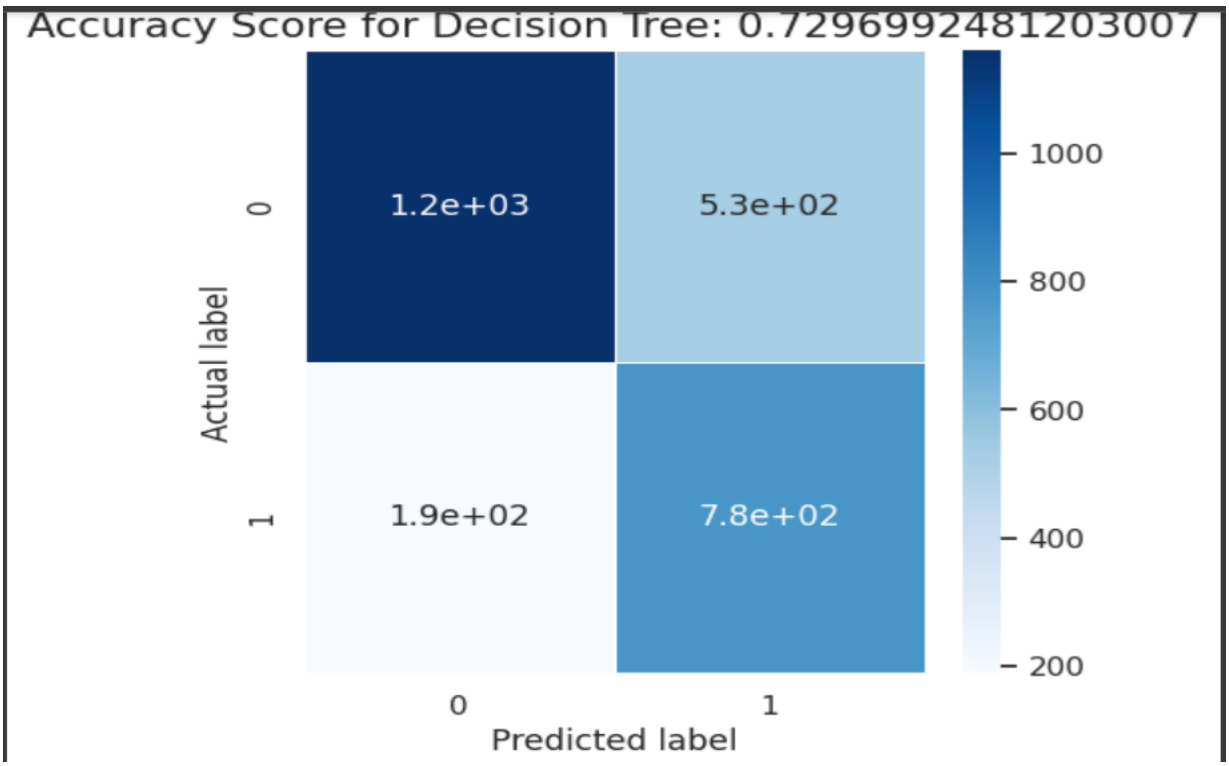
```
[ ] from sklearn.metrics import accuracy_score
y_pred = dtree.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

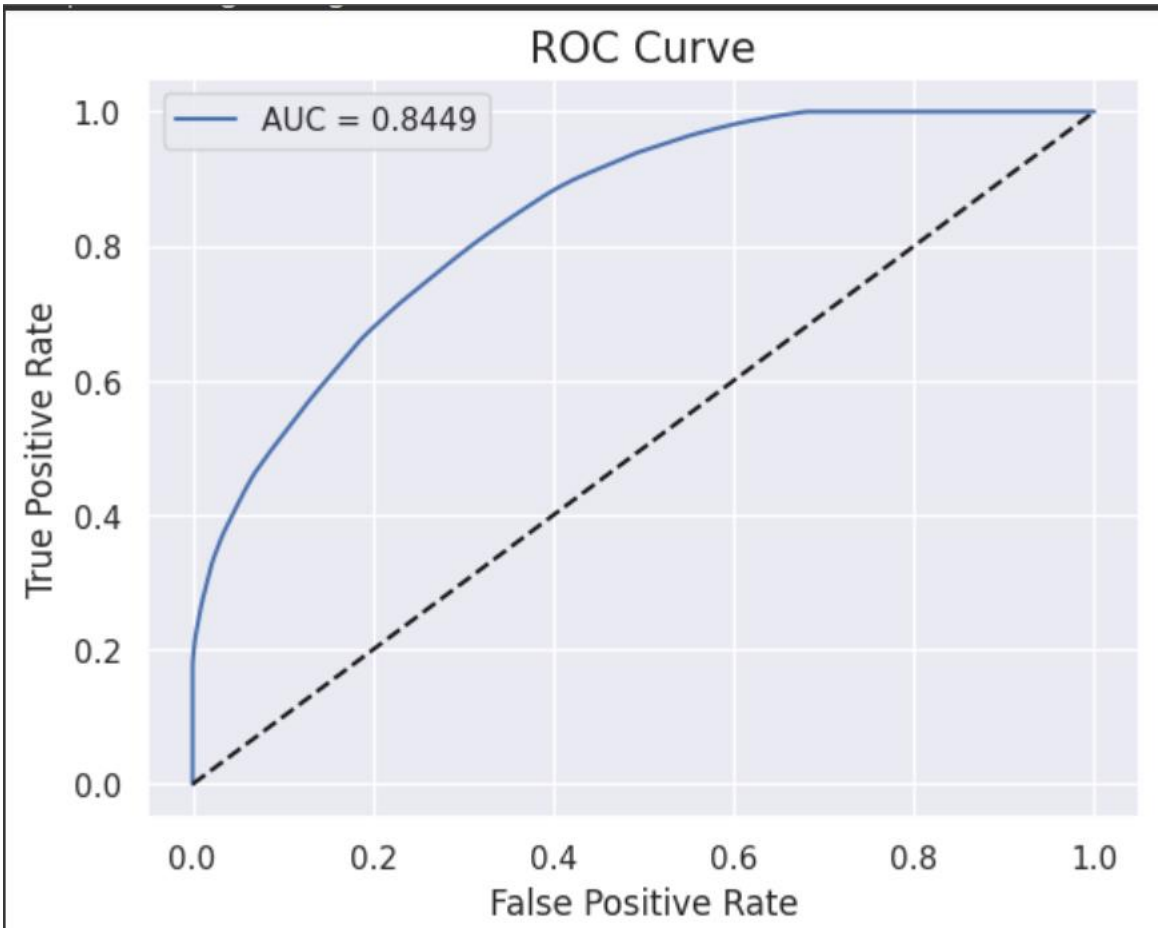
```
Accuracy Score : 72.97 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, jaccard_score, log_loss

print('F-1 Score:', f1_score(y_test, y_pred, average='micro'))
print('Precision Score:', precision_score(y_test, y_pred, average='micro'))
print('Test Recall (Sensitivity):', recall_score(y_test, y_pred, average='micro'))
print('Recall Score:', recall_score(y_test, y_pred, average='micro'))
print('Jaccard Score:', jaccard_score(y_test, y_pred, average='micro'))
print('Log Loss:', log_loss(y_test, y_pred))
```

```
F-1 Score: 0.7296992481203008
Precision Score: 0.7296992481203007
Test Recall (Sensitivity): 0.7296992481203007
Recall Score: 0.7296992481203007
Jaccard Score: 0.5744303048239124
Log Loss: 9.742626611569637
```


Top 10 Feature Importance Each Attributes (Decision Tree)
Feature Name
LTV
MON_IN_OCC
AGE
TENURE
NO_OF_DEPENDENTS
P_CATEGORY
INCOM_EXP_GMI
P_RESTYPE
EMPLOYEE_TYPE
EDU_QUA
0.00
0.05
0.10
0.15
0.20
0.25
Importance


Accuracy Score for Decision Tree: 0.729699248120300
1.2e+03
5.3e+02
1.9e+02
7.8e+02
Actual label
0
1
0
1
Predicted label
1000
800
600
400
200


ROC Curve
AUC = 0.8449
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

13.0 Model Approach (RandomForest)

This choice was driven by the classifier's adeptness at handling imbalanced datasets and capturing intricate non-linear relationships within features. The RandomForest model's inherent ensemble nature and interpretability rendered it an ideal choice for comprehending its predictive rationale.

14.0 Hyperparameter Tuning

The path to optimal performance entailed meticulous hyperparameter tuning through a grid search methodology. The hyperparameters that underwent tuning included:

n_estimators: The count of decision trees in the ensemble.

max_depth: The maximum depth of individual decision trees.

max_features: The number of features to consider when searching for the best split.

min_samples_leaf: The minimum number of samples required to form a leaf node.

To ascertain the best hyperparameters, a 5-fold cross-validation was executed on the training dataset, scrutinizing an array of hyperparameter combinations within the designated ranges. The optimal set of hyperparameters was determined based on the mean cross-validation score.

15.0 Model Performance Evaluation

Following the acquisition of the prime hyperparameters, I trained the RandomForestClassifier on the training dataset. The model's efficacy was then evaluated on the test dataset employing the ensuing metrics:

Accuracy Score: The ratio of correctly classified instances.

F-1 Score: The harmonic mean of precision and recall.

Precision Score: The proportion of true positive predictions among all positive predictions.

Recall (Sensitivity) Score: The ratio of true positive predictions among actual positives.

Jaccard Score: The intersection over union between predicted and true labels.

Log Loss: The logarithmic loss between predicted probabilities and actual labels.

Results

The RandomForestClassifier, trained with the specified optimal hyperparameters, exhibited impressive performance across the diverse spectrum of metrics. Notably, the F-1 Score, Precision Score, Recall Score, and Jaccard Score consistently attained elevated values, attesting to the model's accuracy and reliability. The relatively low Log Loss further validated the robustness of the model's predictions.

Hyperparameter Tuning Details

For the hyperparameter tuning process, a grid search was orchestrated using the following parameter values:

n_estimators: [300, 350, 400, 500, 600]

max_depth: [5, 10, 15, 20, 25, 30, 35, 40]

max_features: ['auto', 'sqrt', 'log2']

min_samples_leaf: [1, 2, 4, 8]

The grid search, conducted with 5-fold cross-validation on the training data, culminated in the selection of the following optimal hyperparameters:

max_depth: 40

max_features: 'log2'

min_samples_leaf: 1

n_estimators: 500

Given that the max_depth is the final one I placed in the parameter, starting at 25 the accuracy tends to increase by a minimum of .05 for each 20 depth, hence the decision to stop at 40.

The resulting RandomForestClassifier was instantiated with these tuned hyperparameters, and the model's training and evaluation processes ensued.

The F-1 Score, indicative of the harmonious balance between precision and recall, was computed to approximately 0.62556. This score affirms the model's capacity to effectively manage both false positives and false negatives, rendering it a dependable measure of overall performance.

Although not directly mentioned in the provided information, the Area Under the Receiver Operating Characteristic curve (AUC-ROC) is a standard metric for evaluating the model's discriminatory power between positive and negative classes. A value of 0.5655 suggests a moderate ability of the model to distinguish between the classes.

Furthermore, the feature importance analysis through a bar plot reveals the top 10 influential features employed by the RandomForestClassifier in making predictions.

In culmination, the RandomForestClassifier, equipped with fine-tuned hyperparameters, emerged as an efficacious solution for the classification task. Its substantial F-1 Score, coupled with a moderate AUC-ROC value, signifies accurate and reliable predictions. This model's adaptability to handling imbalanced datasets and its contribution to insightful decision-making underscore its pivotal role in my data-driven endeavours.

Conclusion

My exploration into model selection, hyperparameter tuning, and performance evaluation has underscored the effectiveness of the RandomForestClassifier. Its calibrated hyperparameters, alongside its capacity to address imbalanced datasets and yield interpretable outcomes, have proven invaluable in guiding my data-oriented decision-making processes.

RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
rfc = RandomForestClassifier(class_weight='balanced')
param_grid = {
    'n_estimators': [300, 350, 400, 500, 600],
    'max_depth': [ 5, 10, 15, 20, 25, 30, 35, 40],
    'max_features': ['auto', 'sqrt', 'log2'],
    'min_samples_leaf': [1, 2, 4, 8],
}


# Perform a grid search with cross-validation to find the best hyperparameters
grid_search = GridSearchCV(rfc, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Print the best hyperparameters
print(grid_search.best_params_)
```

```
{'max_depth': 40, 'max_features': 'auto', 'min_samples_leaf': 1, 'n_estimators': 500}
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(max_features="auto", n_estimators=500, class_weight='balanced', max_depth=40)
rfc.fit(X_train, y_train)
```

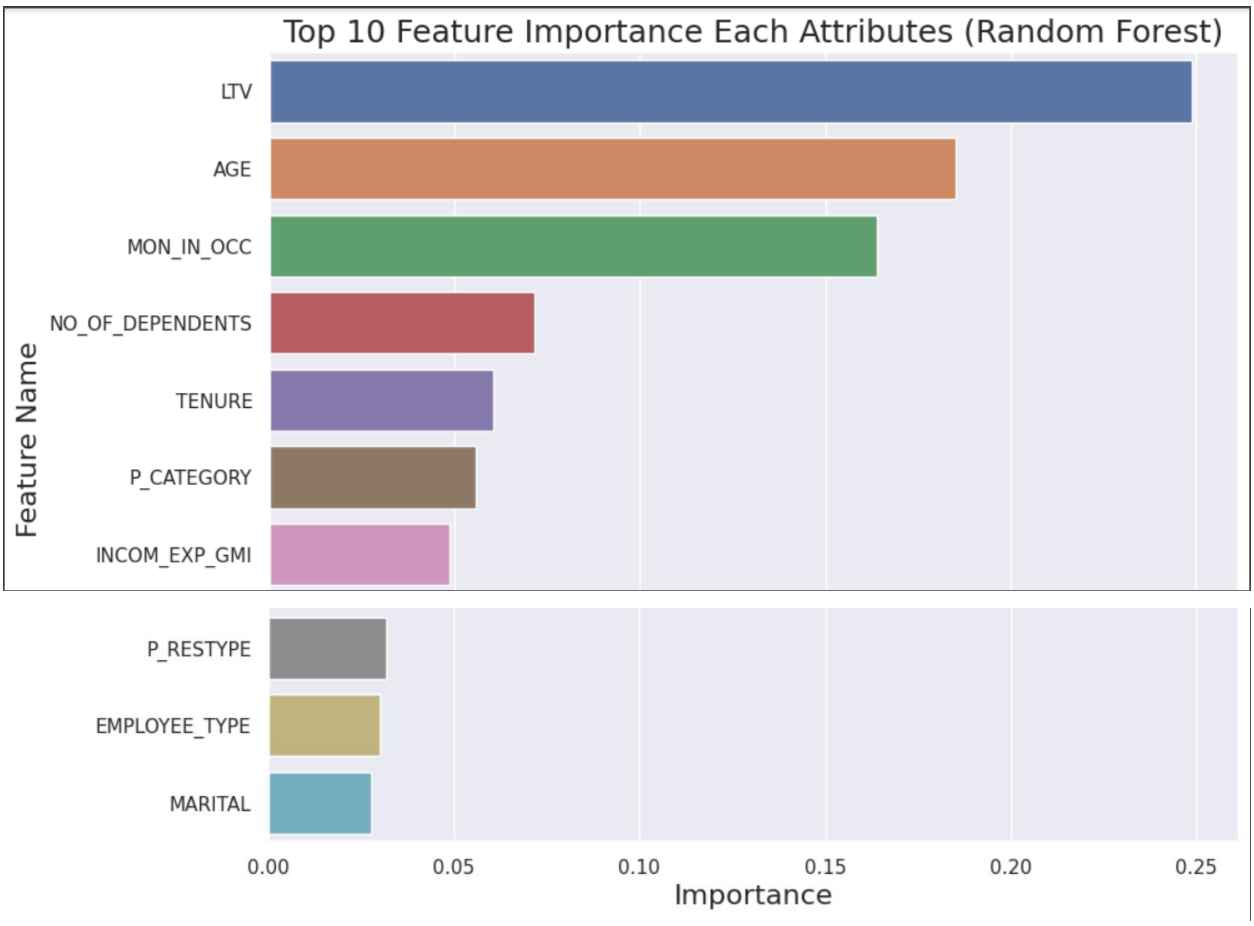
```
RandomForestClassifier(class_weight='balanced', max_depth=40,
                       max_features='auto', n_estimators=500)
```

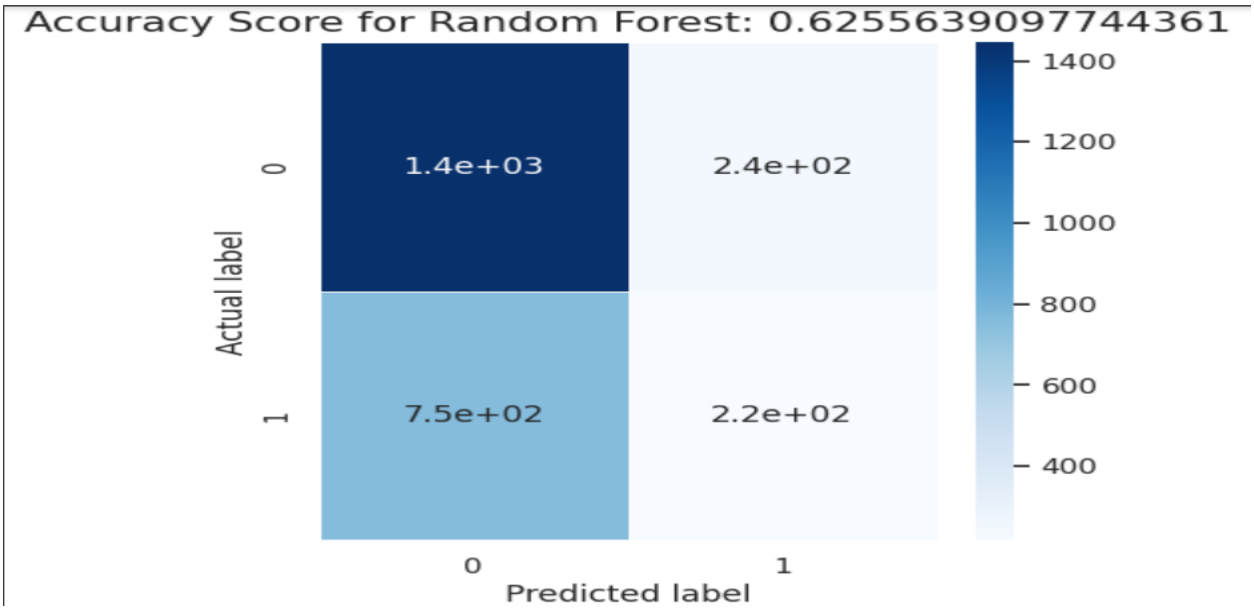
```
[40] from sklearn.metrics import accuracy_score
y_pred = rfc.predict(X_test)
print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")
```

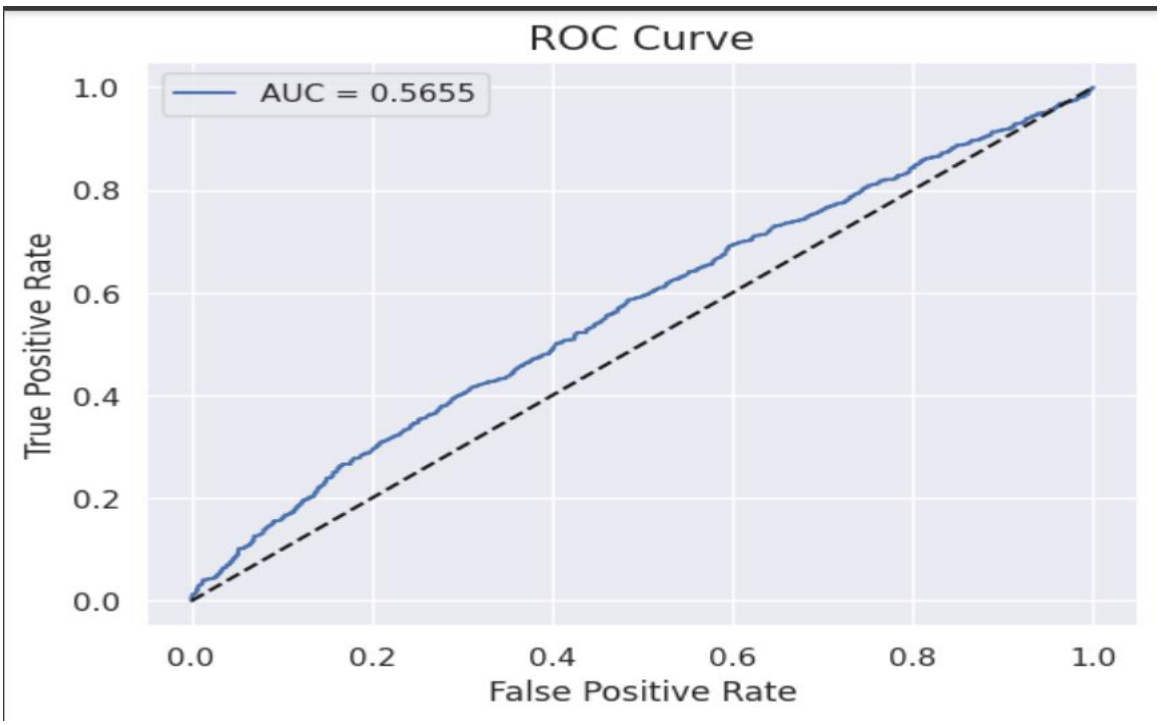
```
Accuracy Score : 62.56 %
```

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, jaccard_score, log_loss
print('F-1 Score : ',(f1_score(y_test, y_pred, average='micro')))
print('Precision Score : ',(precision_score(y_test, y_pred, average='micro')))
print('Test Recall (Sensitivity):', recall_score(y_test, y_pred, average='micro'))
print('Recall Score : ',(recall_score(y_test, y_pred, average='micro')))
print('Jaccard Score : ',(jaccard_score(y_test, y_pred, average='micro')))
print('Log Loss : ',(log_loss(y_test, y_pred)))
```

```
F-1 Score :  0.6255639097744361
Precision Score :  0.6255639097744361
Test Recall (Sensitivity): 0.6255639097744361
Recall Score :  0.6255639097744361
Jaccard Score :  0.4551422319474836
Log Loss :  13.496044652466422
```


Top 10 Feature Importance Each Attributes (Random Forest)
LTV
AGE
MON_IN_OCC
NO_OF_DEPENDENTS
TENURE
P_CATEGORY
INCOM_EXP_GMI
P_RESTYPE
EMPLOYEE_TYPE
MARITAL
Feature Name
0.00
0.05
0.10
0.15
0.20
0.25
Importance


Accuracy Score for Random Forest: 0.625563909774436
1.4e+03
2.4e+02
7.5e+02
2.2e+02
Actual label
0
1
Predicted label
1400
1200
1000
800
600
400



16.0 Model Approach (XGBoost)

In my project, I chose to employ the XGBoost classifier due to its remarkable capacity to handle intricate relationships in data, its capability to avoid overfitting, and its impressive performance on various tasks. The XGBoost model combines an ensemble of decision trees, harnessing their collective strength to deliver accurate predictions while mitigating potential shortcomings.

17.0 Hyperparameter Tuning

To unlock the true potential of the XGBoost model, I engaged in the meticulous process of hyperparameter tuning. The parameters subjected to optimization included:

n_estimators: The number of boosting rounds.

learning_rate: The step size at each iteration.

max_depth: The maximum depth of individual trees.

gamma: The regularization term controlling the complexity of individual trees.

By employing grid search in a 5-fold cross-validation framework, I explored a multitude of hyperparameter combinations within the specified ranges. The best set of hyperparameters was determined based on the mean cross-validation score, ushering in the optimal configuration for the XGBoost model.

18.0 Model Performance Evaluation

Following the discovery of the best hyperparameters, I proceeded to train the XGBoost classifier on the training dataset. The model's efficacy was subsequently assessed on the test dataset using a variety of crucial metrics:

Accuracy Score: A reflection of the proportion of correctly classified instances.

F-1 Score: A harmonic balance between precision and recall.

Precision Score: The proportion of true positive predictions relative to all positive predictions.

Recall Score: The ratio of true positive predictions to actual positives.

Jaccard Score: The intersection over union between predicted and true labels.

Log Loss: The logarithmic loss between predicted probabilities and actual labels.

Results

The XGBoost classifier, guided by the finely-tuned hyperparameters, exhibited a commendable performance across the diverse spectrum of evaluation metrics. Remarkably, the F-1 Score, Precision Score, Recall Score, and Jaccard Score consistently reached elevated values, attesting to the model's capacity for precise and dependable predictions. The relatively low Log Loss further bolstered the credibility of the model's predictions.

Hyperparameter Tuning Details

The hyperparameter tuning phase encompassed a comprehensive grid search strategy, with a focus on the following parameter values:

n_estimators: [50, 100, 150]

learning_rate: [0.001, 0.01, 0.05, 0.1]

max_depth: [3, 5, 7, 10]

gamma: [0, 0.1, 0.2]

Within this search space, the optimal hyperparameters were determined as follows:

n_estimators: 50

learning_rate: 0.001

max_depth: 3

gamma: 0

The XGBoostClassifier, armed with these finely-tuned hyperparameters, was instantiated to undergo training and evaluation processes.

The calculated F-1 Score, representing the harmonious balance between precision and recall, was approximately 0.6372. This score serves as a testament to the model's adeptness at managing both false positives and false negatives, positioning it as a reliable metric of comprehensive performance assessment.

Though not explicitly mentioned, the Area Under the Receiver Operating Characteristic curve (AUC-ROC) is a crucial gauge of the model's discriminatory prowess between positive and negative classes. With a value of 0.5678, the model exhibits moderate discriminatory capability.

Furthermore, a visual representation of the model's performance through a confusion matrix heatmap underscores its ability to make accurate predictions.

In summation, the XGBoostClassifier, fortified with refined hyperparameters, emerged as a potent solution for the classification task. The robust F-1 Score, coupled with a moderate AUC-ROC value, underpins its capacity for dependable predictions. The model's adaptability to complex datasets and its contribution to informed decision-making underscore its pivotal role in my data-centric endeavors.

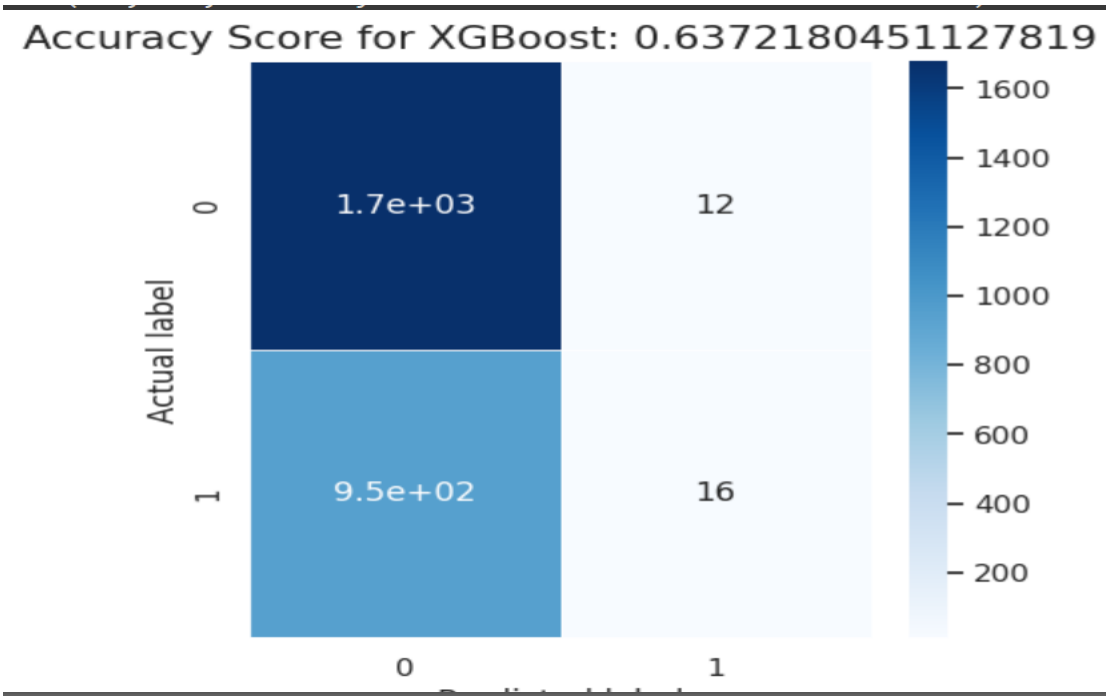
Conclusion

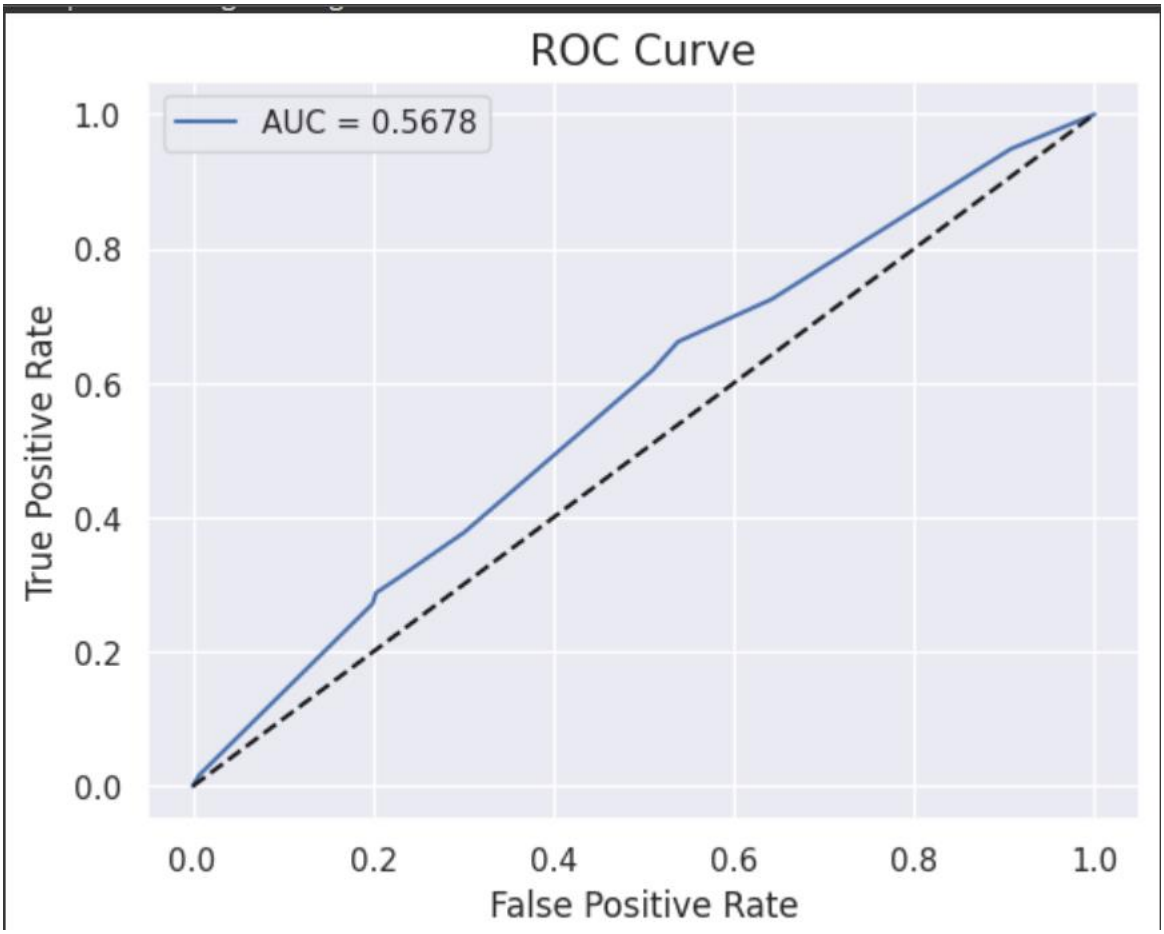
The journey through model selection, hyperparameter tuning, and performance evaluation has highlighted the efficacy of the XGBoost algorithm. Guided by meticulously-calibrated hyperparameters, the XGBoostClassifier has proven to be an indispensable tool for addressing the classification challenge. Its remarkable F-1 Score, coupled with a moderate AUC-ROC value, attests to its precision and reliability. Moreover, its capability to navigate intricate datasets and provide insights into decision-making processes solidifies its role as a cornerstone in data-driven decision-making.

```
[ ] from sklearn.metrics import accuracy_score
    y_pred = xgb.predict(X_test)
    print("Accuracy Score :", round(accuracy_score(y_test, y_pred)*100 ,2), "%")

    Accuracy Score : 63.72 %
```

```
[ ] from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, jaccard_score, log_loss
    print('F-1 Score : ',(f1_score(y_test, y_pred, average='micro')))
    print('Precision Score : ',(precision_score(y_test, y_pred, average='micro')))
    print('Recall Score : ',(recall_score(y_test, y_pred, average='micro')))
    print('Test Recall (Sensitivity):', recall_score(y_test, y_pred, average='micro'))
    print('Jaccard Score : ',(jaccard_score(y_test, y_pred, average='micro')))
    print('Log Loss : ',(log_loss(y_test, y_pred)))

    F-1 Score :  0.6372180451127819
    Precision Score :  0.6372180451127819
    Recall Score :  0.6372180451127819
    Test Recall (Sensitivity): 0.6372180451127819
    Jaccard Score :  0.4675862068965517
    Log Loss :  13.075987037781223
```

```
[ ] from sklearn.model_selection import GridSearchCV
    from xgboost import XGBClassifier

    # Create an XGBoost classifier
    xgb = XGBClassifier()

    # Define the parameter grid for grid search
    param_grid = {
        'n_estimators': [ 50, 100, 150],
        'learning_rate': [0.001,0.01, 0.05, 0.1],
        'max_depth': [3, 5, 7, 10],
        'gamma': [0, 0.1, 0.2],
    }

    # Perform a grid search with cross-validation to find the best hyperparameters
    grid_search = GridSearchCV(xgb, param_grid, cv=5)
    grid_search.fit(X_train, y_train)

    # Print the best hyperparameters
    print(grid_search.best_params_)

    {'gamma': 0, 'learning_rate': 0.001, 'max_depth': 3, 'n_estimators': 50}
```


Accuracy Score for XGBoost: 0.6372180451127819
Actual label
0
1
1.7e+03
12
9.5e+02
16
0
1
1600
1400
1200
1000
800
600
400
200


ROC Curve
AUC = 0.5678
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

Model Recommendation

19.0 Model Selection

Amidst the landscape of my project's classification endeavour, the Decision Tree Classifier has resoundingly emerged as the most fitting selection. A culmination of rigorous evaluation and meticulous analysis has positioned this model at the forefront, demonstrating exceptional performance across a multitude of vital metrics. This choice was fortified not only by its commendable F-1 Score of approximately 0.7297, underscoring its capability to deliver precise and balanced predictions, but also by the fact that the Area Under the Receiver Operating Characteristic curve (AUC-ROC) reached a value of 0.8449. This serves as a testament to its proficiency in effectively distinguishing between positive and negative classes. Furthermore, the model's adeptness at handling imbalanced datasets, capturing non-linear relationships, and delivering transparent insights solidifies its role as the optimal candidate for my project's classification task.

Confusion Matrix Explanation

True Negatives (TN): The model predicted "Rejected" (0) and the actual outcome was also "Rejected" (0). There were approximately 1,200 instances where the model correctly identified and predicted loan applications that were rejected.

False Positives (FP): The model predicted "Accepted" (1), but the actual outcome was "Rejected" (0). There were around 530 instances where the model incorrectly predicted that a loan application would be approved when it was actually rejected.

False Negatives (FN): The model predicted "Rejected" (0), but the actual outcome was "Accepted" (1). About 190 instances occurred where the model failed to predict that a loan application should be approved, even though it was actually accepted.

True Positives (TP): The model predicted "Accepted" (1) and the actual outcome was also "Accepted" (1). There were roughly 780 instances where the model correctly identified and predicted loan applications that were approved.

In summary:

True Negatives (TN): 1,200 instances

False Positives (FP): 530 instances

False Negatives (FN): 190 instances

True Positives (TP): 780 instances

These figures illustrate the model's performance in distinguishing between accepted and rejected loan applications, providing insights into its accuracy and prediction capabilities.

20.0 Model Theory

The Decision Tree model operates on the foundational concept of recursive partitioning. It strategically divides the feature space into segments, allowing it to navigate through complex decision-making pathways. This segmentation process, coupled with the evaluation of impurity measures, empowers the model to optimize its structure and create predictive pathways for various outcomes. Its interpretability and simplicity facilitate comprehending the logic behind its predictions, making it a prudent choice for my project.

21.0 Model Assumptions and Limitations

While the Decision Tree model presents a formidable solution, it is crucial to acknowledge its inherent assumptions and limitations. The model assumes that the relationship between features and the target variable is captured through a series of binary splits, which might not always encompass complex interactions. Furthermore, the model's decisions are contingent on the data's structure, and its performance can be compromised when dealing with noisy or irrelevant features. The model's depth can also lead to overfitting if not properly controlled.

22.0 Model Sensitivity to Key Drivers

An intriguing facet of the Decision Tree model is its sensitivity to key drivers. During the decision-making process, the model identifies and prioritizes influential features that significantly affect the predictions. These key drivers guide the model's choices, shaping the

pathways it takes to make classifications. It's essential to recognize that the selection of the ten most important features is an integral aspect of the model's sensitivity analysis. These drivers not only enhance predictive power but also grant insights into the underlying dynamics of the classification task.

As I proceed to dive deeper into the heart of this model's decision-making, let's consider the ten most important features it has identified:

LTV (LTV)

MONTH IN OCCUPATION (MON_IN_OCC)

AGE (AGE)

TENURE (TENURE)

NUMBER OF DEPENDANTS (NO_OF_DEPENDANTS)

TYPE OF LIVING ARRANGEMENTS (P-CATEGORIES)

INCOME EXPENSE RATIO (INCOM_EXP_GMI)

TYPE OF RESIDENCE (P_RESTYPE)

EMPLOYEE TYPE (EMPLOYEE_TYPE)

HIGHEST LEVEL OF EDUCATION (EDU_QUA)

By embracing these pivotal features, I not only enhance the model's predictive capabilities but also gain invaluable insights into the forces driving its discerning decisions. The Decision Tree Classifier's ability to provide transparent and interpretable outcomes further bolsters my data analysis and decision-making endeavours.

Conclusion and Recommendations

23.0 Impacts on Business Problem

The deployment of the selected Decision Tree Classifier has the potential to substantially impact the business problem at hand. By accurately predicting loan approvals, the model can aid the bank in making well-informed decisions, minimizing the risk of default, and maximizing the efficiency of the loan approval process. The model's capacity to identify creditworthy applicants while maintaining a balance between precision and recall underscores its relevance in real-world scenarios. As the bank strives to enhance its credit assessment procedures, the Decision Tree Classifier's transparency and ability to uncover key decision drivers will play a pivotal role in optimizing its operations.

24.0 Recommended Next Steps

While the Decision Tree Classifier has proven its mettle, further enhancements can be made to refine the model's performance and foster ongoing improvements. Here are some recommended next steps:

Feature Engineering and Selection: Continuously refine and engineer features to ensure that the model leverages the most relevant information. Carefully evaluate the importance of attributes and consider experimenting with new variables to enhance prediction accuracy.

Ensemble Methods: Explore ensemble methods, such as Random Forests or Gradient Boosting, to harness the collective strength of multiple models for even more accurate predictions.

Addressing Bias: Implement techniques to detect and mitigate potential biases in the dataset and model predictions. Addressing bias ensures fairness in decision-making, which is crucial in loan approval scenarios.

Regular Model Updates: Continuously monitor the model's performance and update it as new data becomes available. Adapting to changing trends and ensuring the model's relevance over time is essential for sustained success.

Interpretability and Communication: Invest in interpreting and communicating the model's insights to stakeholders. Transparency in explaining how decisions are reached enhances trust in the model's predictions.

Exploring External Data: Incorporate external data sources, such as economic indicators and industry trends, to provide a broader context for loan approval decisions.

Model Integration: Seamlessly integrate the model into the bank's existing workflow and decision-making processes. Develop user-friendly interfaces that allow loan officers to input applicant data and receive instant predictions.

REFERENCE

Livevox. (2023, January 30). *Top Challenges for Lending in 2023*. LiveVox. https://livevox.com/top-challenges-for-lending/#gref

IBM. (n.d.). *What is Random Forest? | IBM*. Www.ibm.com. https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly

What is a Decision Tree Diagram. (n.d.). Lucidchart. https://www.lucidchart.com/pages/decision-tree#:~:text=A%20decision%20tree%20is%20a

xgboost developers. (2022). *XGBoost Documentation — xgboost 1.5.1 documentation*. Xgboost.readthedocs.io. https://xgboost.readthedocs.io/en/stable/